ARTICLE
A Series of Computational Neuroscience Labs Increases Comfort with MATLAB

David F. Nichols
Psychology Department, Roanoke College, Salem, VA 24153.

Computational simulations allow for a low-cost, reliable means to demonstrate complex and often times inaccessible concepts to undergraduates. However, students without prior computer programming training may find working with code-based simulations to be intimidating and distracting. A series of computational neuroscience labs involving the Hodgkin-Huxley equations, an Integrate-and-Fire model, and a Hopfield Memory network were used in an undergraduate neuroscience laboratory component of an introductory level course. Using short focused surveys before and after each lab, student comfort levels were shown to increase drastically from a majority of students being uncomfortable or with neutral feelings about working in the MATLAB environment to a vast majority of students being comfortable working in the environment. Though change was reported within each lab, a series of labs was necessary in order to establish a lasting high level of comfort. Comfort working with code is important as a first step in acquiring computational skills that are required to address many questions within neuroscience.

Key words: MATLAB; Computational simulations; Hodgkin-Huxley equations; Integrate-and-Fire model; Hopfield Memory network

Computational neuroscience simulations have repeatedly been shown to be useful additions to undergraduate neuroscience laboratory sections (Av-Ron et al., 2006; Molitor et al., 2006; Bish and Schleidt, 2008; Stuart, 2009; Schettino, 2014; Lewis, 2014). Reasons for their inclusion have previously been given that they increase student learning and are low-cost and doable at institutions without extensive lab resources (Bish and Schleidt, 2008). They also provide reliable outcomes (Lewis, 2014) though they can incorporate variability to mimic biological realism (Molitor et al., 2006; Schettino, 2014). Computer simulations have been created previously for education purposes and range from being professionally done and distributed but closed off to direct manipulation by the end user (e.g., Neurons in Action; Stuart, 2009), to being made by the end user and produced for free sharing with a polished graphical user interface (e.g., NeuroLab; Schettino, 2014), to being simple, direct, code-based and open to expansion and specialization (e.g., the current set of simulations). Each of these and others created have a lot to offer students and are useful in different ways. The current paper does not seek to distinguish between them regarding overall usefulness. Rather, the current study addresses the concern that computational simulations need to have a limiting graphical user interface in order for students to be comfortable using them. This will be done by demonstrating changes in students' comfort levels working directly with line-editable code.

Computer simulation laboratories have tended towards graphical user interfaces with values for particular variables available in a subset of options (e.g., Davis, 2001; Molitor et al., 2006; Schettino, 2014). Reasons given for this are that they allow for a user-friendly interface (Stuart, 2009) and hide parameters that students may need to determine (Molitor et al., 2006). On the one hand a problem can arise if not enough options are offered. That is, functionality may be restricted such that the simulations are not sufficiently adaptable to the particular needs of a given class. On the other hand, they can become overly burdensome in their display and choices if they offer too many options. The alternative is to provide the end user (possibly best to be considered the course instructor rather than the student) with the opportunity to modify the code to fit the particular needs of a class or assignment. This can be done with line-editable computer programming software languages, such as MATLAB, Octave, Python, C++, and Fortran. A potential downside is that requiring students to edit particular lines of code in order to get programs to replicate the specific conditions for a given assignment question may make them uncomfortable as it is often outside their prior training. A potential upside is that they are afforded a greater opportunity for exploration and a more powerful means for adapting the specific conditions for follow-up projects. Also, there is an increasing need for knowledge of computer programming in the field of neuroscience.

The current set of labs involves the use of MATLAB, a powerful command-line programming language often used in neuroscience research. The exercises are structured to explain the basic parts of the code in a lab protocol and to walk the students through which lines of code need to be modified for which portions of the lab. The students primarily modify the values of a select and specified set of lines of code and then record the output onto a data note sheet. Prior to the lab period ending, students then complete questions pertaining to the simulations during the lab period and as a small group (2-3 students). Students' comfort level was assessed before and after each of the labs.

MATERIALS AND METHODS

Course: *PSYC/NEUR 330: Principles of Neuroscience Laboratory* met once a week for 1.5 hours in the spring 2015 semester as a required companion to the lecture based course that met three times a week for a total of

three hours per week. The lab and the course were taught by different instructors, both out of the psychology department. The course is a requirement within the neuroscience concentration and an elective within the psychology major. It serves as a pre-requisite for *PSYC/NEUR 430: Research Seminar in Neuroscience* and *NEUR 406: Independent Study in Neuroscience*, one of which is required to complete the neuroscience concentration. Both the course and the lab used *Neuroscience: Exploring the Brain, 3rd Ed.*, by Bear, Connors, and Paradiso (2007).

Students: The class began with 24 students (18 females) and concluded with 22 students (16 females). There were 15 psychology majors, three biology majors, two health and exercise science majors, two who were undecided on their major, and two international exchange students without information on major or year. This distribution is similar to other neuroscience related courses taught out of psychology departments (Wolfe, 2009). Of the non-foreign students, two were seniors, four were juniors, 13 were sophomores, and three were freshmen. The two students who dropped were both sophomore psychology majors. Students were assigned to groups of three after filling out a questionnaire on prior classes and which other students they would or would not like to work with. Students remained in these groups throughout the semester.

Software: All computer simulations were run using MATLAB R2013b. MATLAB requires purchase through an educational license in order to be used in classroom settings (http://www.mathworks.com/academia/). The lab code was saved in script files within lab specific folders on laptops provided to the students. The code was also available on a course specific website for download if students needed to complete the exercises outside of class in a computer lab on campus. The MATLAB software environment presents the experimental code in an editor window that can be opened directly by clicking on the correct file in the provided folder. Students are then asked to save the file immediately while renaming it to include the initials of all group members. The file could then be turned in after completion of the lab as part of required participation/data storage credit.

The program is run directly from the editor window and generally results in the creation of a figure window with task specific data curves. From within the figure window students are often provided with relevant information pertaining to the current simulation in text format, such as the input current, and they can determine additional output information using figure tools, such as the data cursor. There exists a separate command window that allows for the output of results, such as the maximum value of a curve, as well as single lines of commands that students can enter in order to access information about variables, such as the membrane leakage resistance for a particular version of the model.

Labs: *Overview* Across the whole semester there were a total of 12 lab periods, 10 that had specific labs the students completed and two involving aspects of their research proposal projects. The student labs consisted of a mixture of computer simulations and also hands-on data collection. This paper focuses on the computer simulation labs that took place during weeks 2, 4, and 8.

The timing of the labs corresponded with the timing of the course content with a particular emphasis made to perform the labs prior to the relevant information being included on the course exams. To that end, simulations of the Hodgkin-Huxley equations were done during lab 2 and took place prior to exam 1, simulations of the Integrate-and-Fire model were done during lab 4 and took place prior to exam 2, and simulations of Hopfield Memory Networks were done during lab 8 and took place prior to exam 3. The duration of the labs varied based on the amount of material covered but also based on the speed at which particular student groups were able to work through the required material. In general, the completion of the protocol was timed to take from 30 to 50 minutes with an additional 15 to 30 minutes required to complete group questions, leaving sufficient time to put away the equipment and complete short (~5 minutes) pre- and post-lab surveys in the 90 allotted minutes. The primary focus, structure, and group exercises of each of the three labs will be discussed below.

Hodgkin-Huxley Model The Hodgkin-Huxley equations (Hodgkin and Huxley, 1952) were used to focus on the concepts of spike thresholds, spike rate, and how ion-specific membrane conductances relate to the different phases of an action potential. This material is often covered early in introductory neuroscience courses (it is Chapter 4 in the course textbook). It was done the week after the initial lab that used a hands-on demonstration of action potential spiking with a Spiker Box (https://backyardbrains.com/; Marzullo and Gage, 2012) which had a focus of spike rates and action potentials as electrical phenomena. Students were provided with a fully functioning set of code to use in the labs and a written protocol that they were required to read prior to coming to lab (see supplementary materials). Students were initially introduced to the code by opening up the relevant file and then running the program.

The protocol walked students through the process of modifying one or two variables at a time in order to visualize the effect on spiking behavior of the model neuron. For spike thresholds, students were instructed to change the level of input current to a value well below threshold (e.g., “Change line 8 to ‘I=0.01’; and hit run.”) that would result in a time course of the membrane voltage without a clear spike (see Figure 1*A*). They were then instructed to increase the input current incrementally until they reached a current level for which a single spike occurred (see Figure 1*B*) and to record that value on their data note sheet. Following this they were instructed to continue to increase the input current incrementally until a second spike occurred (see Figure 1*C*) and record that value on their data note sheet. Though the instructions allowed for an answer to be found in an algorithmic manner by repeatedly increasing by a small step size, students were free to locate the threshold values by jumping the

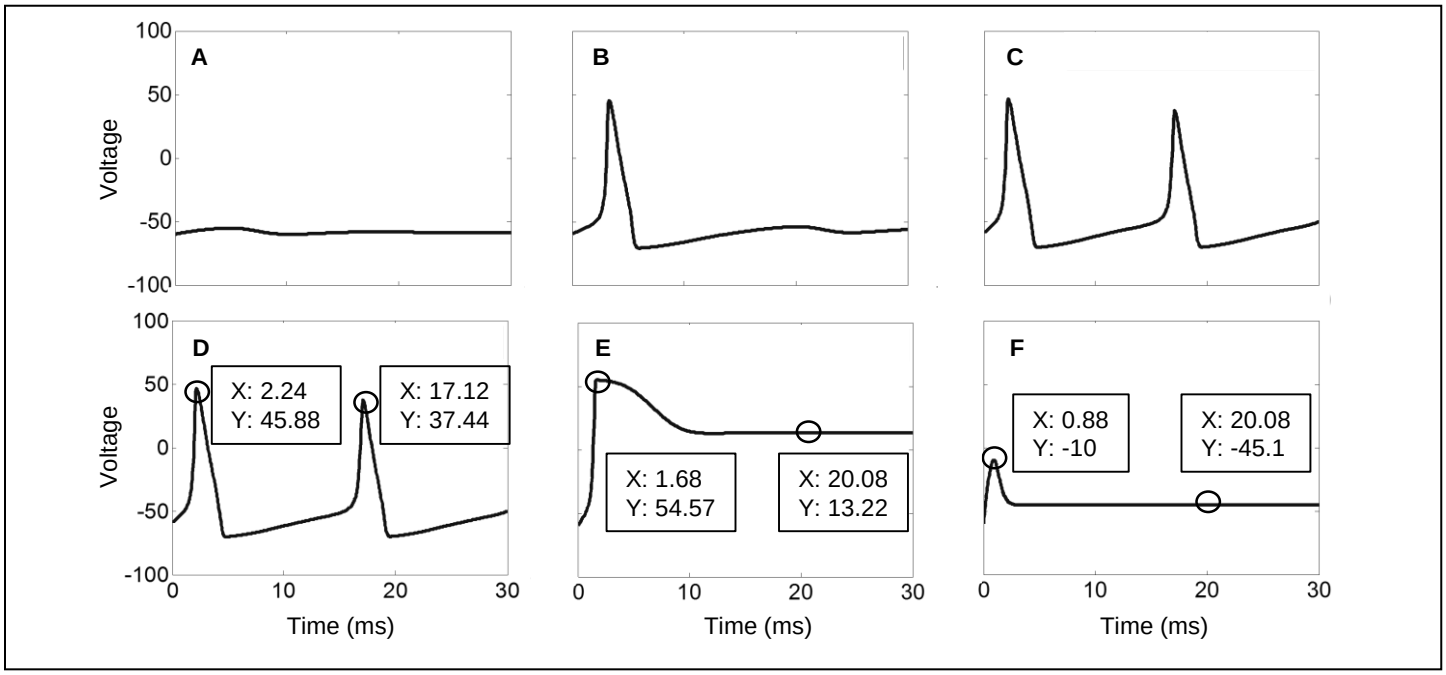

Figure 1. Representative outputs for different components of the Hodgkin-Huxley lab. The line each graph represents the membrane potential for the first 30 ms of simulation. *A*. For subthreshold levels of input current (e.g., 0.02), there is no action potential. *B*. For higher levels of input current (e.g., 0.05) there is only a single action potential. The threshold for the first action potential is at a somewhat lower level. *C*. For even higher levels of input current (e.g., 0.10), there are repetitive action potentials at a constant firing rate. *D*. The timing of the action potentials can be found using the data cursor (represented by the circles) based on the X value of the data plot. *E*. The effects of minimizing the permeability of the membrane to potassium only can be seen by recording the membrane voltage (Y) near the peak level and then at the stabilized level. *F*. The effects of minimizing the permeability of the membrane to sodium only can be seen by comparing the membrane potential (Y) near the peak level to the peak level in the original equations (*D*).

current levels by varying amounts and often started doing so as they progressed through the exercises.

In addition to visualizing qualitative changes in the model output, students engaged in exercises to record quantitative changes in output. For spike rates, the input current was set at a value above the threshold for generating two spikes and then the data cursor component of the figure window was first used to locate the timing of one spike, then the timing of the other spike (see Figure 1*D*). These values were recorded on the data note sheet for later conversion to firing rates. To complete the exercise, students were instructed to increase the input current to higher levels in order to determine a maximum firing rate for the model. For ion-specific membrane conductance, students were required to modify one of the lines of code specifying the membrane conductance for a particular ion (e.g., "Change line 9 to 'gbarNa=0'") and then change the input current to pre-specified values. Quantitative measures of the effect this had on particular phases of the action potential were taken again using the data cursor but this time based on the initial rise in membrane potential and later settling value for when there was no potassium conductance (see Figure 1*E*) and when there was no sodium conductance (see Figure 1*F*).

Integrate-and-Fire Model A simple version of an integrate and fire neuron (Abbot, 1999) was used to focus on the concepts of post-synaptic potentials and temporal summation in addition to reviewing concepts of firing rates and action potentials as electrical phenomena. This lab was done the week after a hands-on lab with an emphasis on firing rate and conduction velocity using human electromyography and corresponded to content covered in Chapters 4 and 5 of the textbook. There was intentional repetition of quantitative and qualitative data analysis techniques involving recording the time between spikes for firing rates and thresholds of initial spikes as these techniques were considered key concepts and foundational research skills.

Specific code was created to explore the concepts of excitatory post-synaptic potentials (EPSPs) and inhibitory post-synaptic potentials (IPSPs) separately. This was done in order to create topic specific graphs that focused in on the concept of interest for a specific portion of the exercise. For EPSPs, the code allowed for the independent specification of the magnitude, i.e., input current, and timing, i.e., relative delay, of two short duration (2 ms) inputs. This was used to demonstrate how two very large EPSPs would not require temporal summation in order to reach threshold (see Figure 2*A*), but that two smaller EPSPs would not result in an action potential (see Figure 2*B*) unless they were sufficiently close together in time (see Figure 2*C*).

The variables of the model, such as capacitance and leakage resistance, were also modified to demonstrate the role they play in temporal summation. Students recorded the threshold level for the input current or time delays that

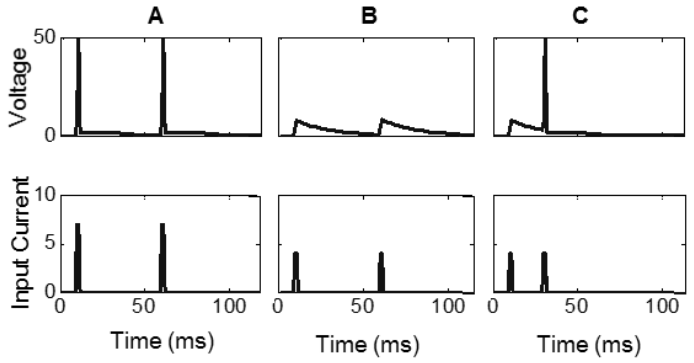

Figure 2. Representative outputs for the temporal summation component of the Integrate-and-Fire lab. The line in the top row of graphs represents the membrane voltage for the first 100 ms of simulation and the line in the bottom row of graphs corresponds to the magnitude and time course of the EPSPs. *A*. An action potential occurs after each high magnitude EPSP (e.g., 7). *B*. For medium levels of EPSP (e.g., 4) at the same temporal distance apart as in (*A*), no action potentials occur. *C*. For medium levels of EPSPs (e.g., 4) that occur closer together in time, a single action potential occurs due to temporal summation.

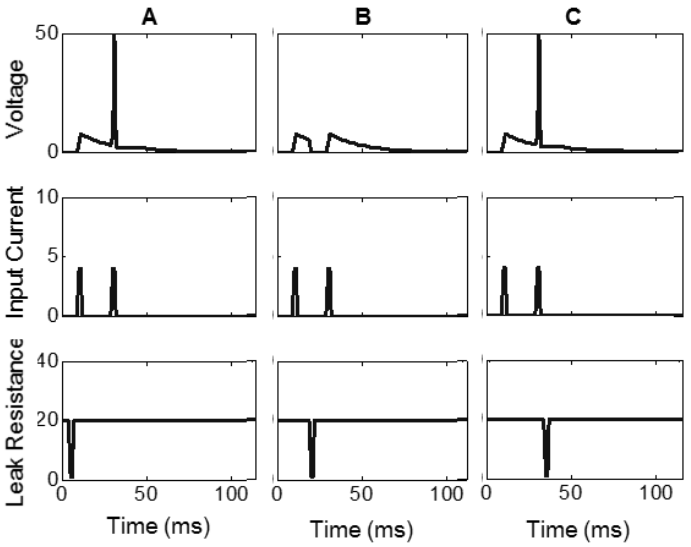

Figure 3. Representative outputs for the shunting inhibition component of the Integrate-and-Fire lab. The line in the top row of graphs represents the membrane voltage for the first 100 ms of simulation, the line in the middle row of graphs represents the magnitude and time course of the EPSPs, and the line in the bottom row of graphs represents the time course of the IPSPs as a complete shunting of all membrane voltage. *A*. The timing of the IPSP (5 ms) is before the first EPSP (10 ms) and temporal summation is possible. *B*. The timing of the IPSP (20 ms) is in between the first EPSP (10 ms) and the second EPSP (30 ms) and temporal summation does not occur. *C*. The timing of the IPSP (35 ms) is after the second EPSP (30 ms) and temporal summation is possible.

were required to result in action potentials. For IPSPs, the code was primarily set up to manipulate the timing of a transient change in the leakage resistance that served to abolish any buildup of membrane potential towards threshold, i.e., shunting inhibition. This was used to demonstrate that IPSPs that occur prior to the EPSPs (see Figure 3*A*) or after both of the EPSPs (see Figure 3*C*) have no effect on temporal summation, but those that occur between the EPSPs do not allow for temporal summation to occur (see Figure 3*B*). Students systematically changed the variable controlling the time of the IPSP and recorded the range of time for which the IPSPs resulted in an action potential not occurring.

Hopfield Memory Network Model A minimal version of the Hopfield memory network (Hopfield, 1982) was used to demonstrate the principles of neural networks and memory retrieval. It is essentially based on the principles of Hebbian learning in the establishment of connections between nodes of the network such that the nodes that are regularly on or off together within a particular set of memories will have stronger synaptic connections than nodes that are uncorrelated. However, the basic Hopfield memory network is focused to a much greater degree on memory retrieval than memory formation/learning, as the synaptic connections are entirely determined by the set of memories presented to the network and remain constant. The simulations that evolve over time are based on the updating of the states of the nodes from an initial input state towards a stable memory state. Along those lines, the lab itself was primarily focused on determining the robustness of the memory network in the correct retrieval of memories in relation to different levels of input noise. The protocol and lab questions included sections on Hebbian learning in relation to the functioning of the neural network.

Different sets of code were set up that involved different numbers of memories with distinct tasks associated with each number of memories. The initial code had four memories visually represented as black letters on a white background. This code primarily involved walking students through the basics of how the network functioned. This included a demonstration of how small numbers of errors from the initial state (e.g., two out of 25 neurons having reversed on/off states; see Figure 4*A*) resulted in the successful retrieval of the memory in the stable state of the network whereas large numbers of errors from the initial state (e.g., 10 out of 25 neurons having reversed on/off states; see Figure 4*B*) resulted in an incorrect memory state being stabilized. This also included students setting their own criterion for how the performance would be evaluated by setting thresholds that defined whether the model worked for a given level of noise based on the percentage of matching nodes and/or the percentage of exact memory matches. In order to quantitatively measure this, a related but distinct set of code was run that conducted 10,000 simulations (which takes only a few seconds) and reported the overall performance of the network (based on the proportion of matching nodes and the relative frequency of exact matches).

The next set of code had two memories and primarily involved students making educated guesses about how well the memory system would perform based on different pairs of memories (see Figure 5) in relation to what they had thus far learned in the lab in regards to the synaptic connections of the neural network. Students essentially

make their choice based on whether memories will be successfully retrieved more frequently if there are higher or lower amounts of consistent node states between the pairs. The amount of consistent node states is provided for them numerically in the command window in addition to the visual representation shown in a figure window. The final set of code had six memories with the students required to choose a pattern of on/off node states to represent two new memories of their choosing and then enter them into the code at clearly marked locations. Following this the same procedure was conducted as in earlier portions of the exercise whereby the network was simulated 10,000 times and the output was reported in the command window for students to enter into their data note sheets.

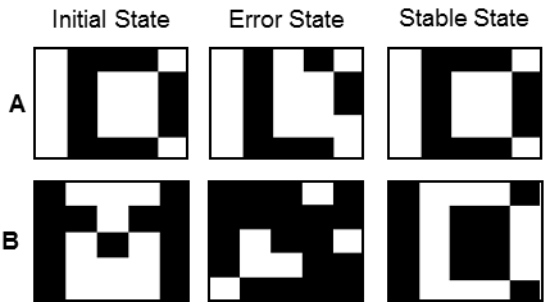

Figure 4. Example outputs of the Hopfield Memory Network lab. Each graph shows the activation for 25 nodes in a 5x5 grid where white indicates a particular node is "on" and black indicates a particular node is "off." The initial state refers to the input prior to noise and is the correct state for the network to end up in if it successfully retrieves the memory. The error state shows how a subset of the nodes are flipped from the initial state and is the actual input to the memory network. The stable state is the output of the memory network. The network clearly worked if the stable state exactly matches the initial state and clearly did not work if a different memory state was retrieved. *A*. The typical performance based on 4 memories and a low level of noise (2 flips) with the stable state matching the initial state. *B*. The typical performance based on 4 memories and a high level of noise (10 flips) with the stable state clearly different from the initial state.

Figure 5. The pattern of network activation for the two memory exercise within the Hopfield Memory Network lab. Students were instructed to pay attention to the relative overlap in the patterns for Pair 1 (D vs. M) and Pair 2 (J vs. C).

Surveys: In order to document the level of how comfortable students were with the MATLAB environment and assess the amount of student learning on certain key concepts, short surveys were passed out and voluntarily completed at the beginning and end of each lab section. The pre-lab surveys each contained three questions with a consistent format. The first two questions pertained to the level of understanding for key concepts to be covered in the lab with responses on a scale from 0 (no understanding) to 10 (complete understanding) with only the extreme values labeled. The last question pertained to how comfortable the students were with the hardware or software utilized in that day's lab with responses on a scale from -5 (very uncomfortable) through 0 (neutral) to 5 (very comfortable) with only those three values labeled. The post-lab survey repeated the three questions from the pre-lab survey and for each of those questions students were also asked how much the current lab influenced their understanding or comfort on a scale from -5 (understand it much less/much less comfortable) through 0 (no change) to 5 (understand it much more / much more comfortable) with only those three values labeled.

The questions pertaining to level of understanding of concepts were lab specific whereas the final question relating to comfort levels on the software was identical across all three labs. For the Hodgkin-Huxley lab, the concept questions were "How well do you understand the concept of thresholds for generating action potentials?" and "How well do you understand the concept of different ions influencing parts of action potentials?". For the Integrate-and-Fire lab, the concept questions were "How well do you understand the concept of thresholds for generating action potentials?", which was a repeat from the Hodgkin-Huxley lab, and "How well do you understand the concept of EPSPs and IPSPs in relation to action potentials?". For the Hopfield Memory Network lab, the concept questions were "How well do you understand the concept of memory storage in neural networks?" and "How well do you understand the concept of retrieval failure in neural networks?". For all three labs, the final question was "How comfortable are you with using MATLAB – a command line computer programming software language?"

Students were informed at the beginning of the semester via email and in the class and reminded repeatedly throughout the semester that participation in the surveys was voluntary and would in no way affect their grades on the labs. Student names were included on the surveys in order to pair the pre-lab surveys with the post-lab surveys and to track changes in responses across labs. However, the surveys were inputted into an Excel sheet without the corresponding student names and the faculty member did not review the responses of the students until after the semester was completed. This data collection was approved by the Internal Review Board of Roanoke College.

RESULTS

There were 23 students in the course for labs 2 and 4 and 22 students for lab 8. All 23 students completed the pre- and post-surveys for lab 2 (Hodgkin-Huxley equations). While 22 students completed the pre-survey for labs 4 (Integrate-and-Fire model) and 8 (Hopfield Memory network), only 21 completed the post-survey. Only 18 students completed all six surveys. Analyses were done with t-tests for individual survey questions based on all students who completed that particular question and Repeated Measures ANOVAs for sets of questions based on only the subset of students who completed all relevant questions. Reports of significance are based on $\alpha=0.05$.

Comfort with MATLAB: As MATLAB was used consistently across all three computer simulation labs, the same question was asked for all six surveys regarding how comfortable the student was with using MATLAB at the time that the survey was taken. Additionally, for all three post-lab surveys, the same question was asked regarding how much they felt that particular lab influenced their comfort level. The level of comfort that students would come into the course with was expected to be fairly low based on anecdotal student reactions to similar labs in a previous semester. A somewhat low initial comfort level would allow for increases or decreases in the level of comfort across the semester. Measuring comfort level across multiple labs also allows for student comfort levels to remain constant or show saturation after a limited number of labs on an individual basis.

95% confidence intervals for the level of self-reported comfort on a scale from -5 (very uncomfortable) through 0 (neutral) to 5 (very comfortable) on each lab survey are shown in the left portion of Figure 6*A*. Clearly evident in the graph is that students on average started off not significantly different from a neutral comfort level for lab 2 ($\bar{x}$=-0.8, t(22)=-1.40, p=0.174) and lab 4 ($\bar{x}$=0.7, t(21)=1.33, p=0.197), but they were comfortable at the start of lab 8 ($\bar{x}$=2.2, t(21)=5.73, p<0.001). By the end of each of the labs, however, the comfort level was significantly above neutral ($\bar{x}$'s>2.1, t's(20)>5.6, all p's<0.001).

In order to assess differences between the labs, a 2x3 Repeated Measures ANOVA was run with the time of the survey (pre, post) and the week of the lab (2, 4, 8) as factors. Mauchly's Test of Sphericity indicated that the assumption of sphericity was violated for both the week of the lab factor (X^2(2)=13.35, p=0.001) and the interaction (X^2(2)=6.61, p=0.037). Therefore, the Lower-bound correction with the degrees of freedom reduced by half will be reported as a highly conservative correction. There was a main effect of time of the survey (F(1,18)=75.553, p<0.001) with post-lab ratings on average 2.14 rating points higher than pre-lab ratings. There was also a main effect of week of the lab (F(1.0,36)=12.729, p=0.002) explained by a linear effect (F(1,18)=21.695, p<0.001) wherein the comfort level increased across the labs. Comfort levels increased from lab 2 to lab 4 by 1.08 rating points (p=0.049) and from lab 4 to lab 8 by 1.00 rating points (p<.001). Furthermore there was an interaction effect (F(1.0,36)=4.867, p=0.041) explained by a linear effect (F(1,18)=14.329, p=0.012) wherein the difference in pre-post comfort level decreased as a function of the week of the lab. The difference in comfort levels for lab 2 was 2.94 rating points, for lab 4 was 2.26 rating points, and for lab 8 was 1.21 rating points.

In addition to the difference in comfort levels between the pre and post surveys, data was also collected on how much students felt that each particular lab influenced their comfort level. As can be seen in the right portion of Figure 6*A*, the students on the whole reported that each lab resulted in a positive change in their comfort level ($\bar{x}$'s>2.6, t's(20)>8.0, all p's<0.001). In order to compare across labs, a one-way Repeated Measures ANOVA was run. Since the assumption of sphericity was again violated

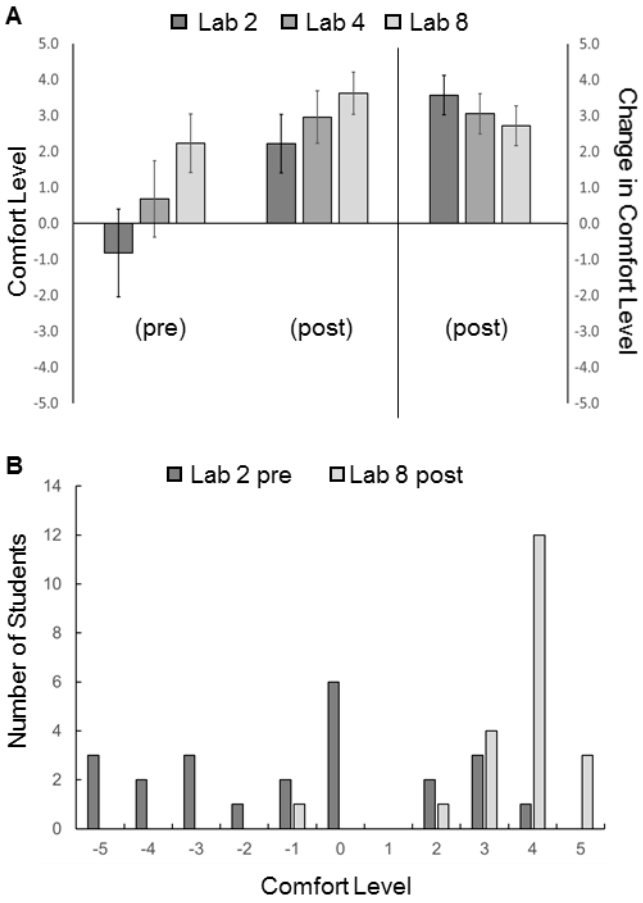

Figure 6. Results of questions pertaining to the self-reported comfort level working with the MATLAB programming language. *A*. The set of bars on the left shows the level of comfort before and after the labs were completed averaged across all students. The scale is from Very Uncomfortable (-5) through Neutral (0) to Very Comfortable (5). The bars on the right show the degree to which students felt that week's lab contributed to their level of comfort with MATLAB. The scale is from Much Less Comfortable (-5) through No Change (0) to Much More Comfortable (5). Error bars indicate the 95% confidence interval for each of the separate survey questions. *B*. Frequency distribution of students across comfort levels before the first computational simulation lab (Lab 2 pre) and at the end of the final computer simulation lab (Lab 8 post). The scale is from Very Uncomfortable (-5) through Neutral (0) to Very Comfortable (5).

(X^2(2)=7.21, p=0.027), the marginally significant main effect of week of the lab (Lower-bound correction: F(1.0,36)=3.814, p=0.067; Sphericity assumed: F(2,36)=3.814, p=0.031) needs to be considered with caution. Similarly, though a linear trend was again indicated (F(1,18)=4.80, p=0.042), neither the change from 3.53 to 3.00 for labs 2 and 4 (p=0.086) nor from 3.00 to 2.68 for labs 4 and 8 (p=0.187) was statistically significant. Overall this indicates that there was a less consistent relationship between the week of the lab and how much students reported that each lab contributed to their comfort level than in the observed difference in the pre- vs. post-lab comfort levels. This could be explained by students believing that each lab improved their comfort levels even as they ran out of room to convey that difference in the pre-

vs. post-lab comfort levels. That is, reporting change after each lab may be helpful in avoiding ceiling effects when just considering differences in the pre- vs. post-lab comfort levels.

The above analysis demonstrates that comfort levels with MATLAB increased within each of the labs and across the sequence of labs. To further illustrate this, the distribution of reported comfort levels for the first survey and last survey are shown in Figure 6*B*. A substantial proportion of the students (11/23) reported being uncomfortable using MATLAB prior to the first computational simulation lab whereas nearly all (20/21) reported being comfortable after the final computational simulation lab. The trend for a decrease in the amount that each week's lab contributed to a student's overall comfort level could then be expected to be due to an increase in the number of students who were comfortable with MATLAB at the start of each lab.

Understanding of concepts: For each of the labs, questions were asked about the level of understanding of two specific key concepts and also about the contribution of that particular lab to their level of understanding. The timing of the labs was such that relevant concepts had already been covered in the lecture portion of the class for labs 2 and 4 but not for lab 8. Correspondingly, the self-reported level of understanding of the two concepts on the pre-surveys was relatively high for labs 2 and 4 (7.0, 7.2, 7.4, and 6.3, respectively, on a scale from 0 to 10) and lower for lab 8 (4.8 and 5.9). Still, there was a significant increase in the rating scales for the pre- to post-lab levels of understanding for all concept questions ($\bar{x}$'s>0.8, t's(20)>2.7, all p's<0.02) and students reported that each lab increased their level of understanding on that concept ($\bar{x}$'s>2.1, t's(20)>7.9, all p's<0.001).

DISCUSSION

Students in neuroscience come from a broad range of majors, many of which do not require computer science courses. Therefore, it is worth considering what factors may contribute to how quickly and to what degree students can become comfortable working with code. Based on the starting and ending levels of comfort, it can be inferred that inexperience working with code is not a hindrance to using it during labs and that a series of labs is likely useful for increasing comfort levels. It is an encouraging sign that students can become comfortable with code-based exercises as computational thinking is a foundational skill required for the next generation of scientists (Wing, 2006).

The rate at which comfort level changes could be influenced by whether labs are completed individually or in groups, with the group size a further contributing factor. Here groups were generally three people though occasionally groups of two were used due to students dropping the course or being absent. Groups were encouraged to alternate which student was primarily in charge of controlling the MATLAB program as the lab was completed, and anecdotally it seemed that controlling the program helped increase student comfort level more than simply watching another student use the program due to an increase in active engagement.

One of the proposed benefits of computational simulation labs is that they can be inquiry-based, hypothesis-driven exercises (Crisp, 2012; Lemons, 2012; Lewis 2014). The current set of labs is largely protocol driven and not inquiry-based, but that is in part due to the small amount of time (90 minutes) allotted to the lab periods. For a longer lab period (2-3 hours), it is entirely conceivable that the first half could be devoted to familiarizing students with the computational models using a protocol and that the second half could be devoted to answering an empirical question of the students' choosing. The neuroscience curriculum at Roanoke College requires a semester long research project, so the inquiry-based training is largely done at that time rather than during the lab component of the introductory level course.

Computational simulations are a worthwhile and helpful component of an undergraduate neuroscience lab course (Lewis, 2014). They are inherently reliable since the results will consistently come out the same when the same values are put into the model equations, which cannot be said for hands-on neuroscience with live animals, though stochastic noise can also be added to the equations in order to approach biological realism. They are valid in so much as they truly represent the principles being simulated, which cannot be guaranteed with hands-on neuroscience when artifacts are possible in the data due to the equipment or experimenter error. A benefit of code-based simulations in relation to compiled simulations is that they are adaptable such that the activities can be easily modified to fit the expectations/concepts covered in a particular class. MATLAB in particular is a powerful yet approachable programming environment with a low cost in comparison to standard hands-on neuroscience equipment, but Python is an adequate alternative for programs without the budget to purchase MATLAB or with faculty already familiar with working in Python. Students are likely to be amenable to becoming familiar and comfortable with whatever coding software is used in the labs. Early exposure can lead to future opportunities to further knowledge using code, such as in undergraduate research projects, graduate school, or jobs.

Supplementary Materials

The MATLAB code, all associated assignment files, descriptions of the assignment files, and links to resources for comparable Python code can be found at the author's faculty page:
https://directory.roanoke.edu/faculty?username=dnichols.

REFERENCES

Abbott (1999) Lapicque's introduction of the integrate-and-fire model neuron (1907). Brain Res Bull 50:303-304.

Av-Ron E, Byrne JH, Baxter DA (2006) Teaching basic principles of neuroscience with computer simulations. J Undergrad Neurosci Educ 4:A40-A52.

Bish JP, Schleidt S (2008) Effective use of computer simulations in an introductory neuroscience laboratory. J Undergrad Neurosci Educ 6:A64-A67.

Crisp KM (2012) A structured-inquiry approach to teaching neurophysiology using computer simulation. J Undergrad Neurosci Educ 11:A132-A138.

Davis MJ (2001) Basic principles of synaptic physiology illustrated by a computer model. Adv Physiol Educ 25:1-12.

Hodgkin AL, Huxley AF (1952) A quantitative description of membrane current and its application to conduction and excitation in nerve. J Physiol 117:500-544.

Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. Proc Natl Acad Sci USA 79:2554-2558.

Lemons ML (2012) Characterizing mystery cell lines: student-driven research projects in an undergraduate neuroscience laboratory course. J Undergrad Neurosci Educ 10:A96-A104.

Lewis DI (2014) The pedagogical benefits and pitfalls of virtual tools for teaching and learning laboratory practices in the biological sciences. The Higher Education Academy: STEM.

Marzullo TC, Gage GJ (2012) The SpikerBox: a low cost, open-source bioamplifier for increasing public participation in neuroscience inquire. PLoS ONE 7:e30837.

Molitor SC, Tong M, Vora D (2006) MATLAB-based simulation of whole-cell and single-channel currents. J Undergrad Neurosci Educ 4:A74-A82.

Schettino LF (2014) NeuroLab: a set of graphical computer simulations to support neuroscience instruction at the high school and undergraduate level. J Undergrad Neurosci Educ 12:A123-A129.

Stuart AE (2009) Teaching neurophysiology to undergraduates using Neurons in Action. J Undergrad Neurosci Educ 8:A32-A36.

Wing JM (2006) Computational thinking. Commun ACM 49:33-35.

Wolfe U (2009) Successful integration of interactive neuroscience simulations into a non-laboratory Sensation & Perception course. J Undergrad Neurosci Educ 7:A69-A73.

Received August 05, 2015; revised September 24, 2015; accepted September 29, 2015.

The author thanks Dr. Darcey Powell for her input on the construction of the surveys and comments on an earlier version of the manuscript.

Address correspondence to: Dr. David F. Nichols, Psychology Department, 221 College Lane, Roanoke College, Salem, VA 24153. Email: dnichols@roanoke.edu