ARTICLE
An Algebra-Based Introductory Computational Neuroscience Course with Lab

Christian G. Fink
Neuroscience Program, Ohio Wesleyan University, Delaware, OH 43015.

A course in computational neuroscience has been developed at Ohio Wesleyan University which requires no previous experience with calculus or computer programming, and which exposes students to theoretical models of neural information processing and techniques for analyzing neural data. The exploration of theoretical models of neural processes is conducted in the classroom portion of the course, while data analysis techniques are covered in lab. Students learn to program in MATLAB and are offered the opportunity to conclude the course with a final project in which they explore a topic of their choice within computational neuroscience. Results from a questionnaire administered at the beginning and end of the course indicate significant gains in student facility with core concepts in computational neuroscience, as well as with analysis techniques applied to neural data.

Key words: computational neuroscience, neural modeling, computer programming, MATLAB

Computational neuroscience (CN) is a fast-growing field. The U.S. BRAIN Initiative represents one example of the need to analyze vast troves of neural data and distill the results to theoretical models of how the brain processes information (Insel et al., 2013). While many courses in computational neuroscience have been developed previously, most (if not all) require multiple prerequisite classes in calculus and computer programming (Dayan and Abbott, 2001; Trappenberg, 2009; Érdi, 2015), which many traditional neuroscience majors do not take. Furthermore, such courses typically focus on theoretical models of how neural systems compute, but do not address methods of analyzing neural data.

The Computational Neuroscience course at Ohio Wesleyan was developed to meet these needs. Students with no prior experience in calculus or programming learn to code in MATLAB and apply this skill to simulate neural processes (using finite difference equations, rather than differential equations) and to analyze real-world neural data. While the lack of technical prerequisites necessarily limits the pace compared to other courses in CN, this CN course provides traditional neuroscience students the opportunity to learn the language and mode of thinking of CN, thus enriching their understanding of neural systems while developing general data analysis skills that are applicable in any laboratory setting.

LEARNING OBJECTIVES

Upon successful completion of this course, students will be able to:

1) explain the fundamental principles of information processing by neural systems (as they are currently understood).
2) model simple neural systems by writing simulation code in MATLAB.
3) analyze various forms of neural data using fundamental computer programming techniques.

COURSE RESOURCES

The course requires three primary resources for students: the textbooks *A Tutorial in Neural Systems Modeling* (by Thomas J. Anastasio) and *MATLAB for Neuroscientists: An Introduction to Scientific Computing in MATLAB* (by Pascal Wallisch and Michael Lusignan), as well as a computer with access to MATLAB for every student in the course. Students may also purchase their own discounted MATLAB student license at http://www.mathworks.com/academia/student_version/. Instructors may access course materials developed by the author at http://chrisfink.xyz/teaching/teaching_resources.html.

COURSE DESIGN

Prerequisites and intended audience

This course is intended for junior- and senior-level neuroscience majors, and is specifically intended to develop breadth of knowledge and skills for majors whose concentration is behavioral, cognitive, or cellular/molecular neuroscience. (Students majoring in Ohio Wesleyan's separate Computational Neuroscience major take a different course, in theoretical neuroscience.) Course prerequisites at Ohio Wesleyan are Introduction to Neuroscience and a low-level statistics course (to ensure some quantitative competency). Optimal class size is eight to ten students. Any more than this, and it becomes difficult for the instructor to keep up with students' questions as they work through in-class and lab exercises. Class size could easily be expanded, however, with the addition of a TA to help answer questions.

Structure

The two fundamental aspects of the course are separated between the classroom (meeting for 50 minutes three times per week) and lab (meeting once for three hours every week), with the classroom portion exploring neural models through simulation and the lab portion covering data analysis techniques. Students are expected to dedicate 8-10 hours to the course per week outside of class time. Because no previous programming experience is assumed, the first three weeks of both the classroom

and lab are spent building competence in fundamental programming skills (*for* loops, *if* statements, relational operators, functions, etc.), motivating these topics with as many examples from neuroscience as possible. MATLAB is used as the programming language due to its relatively shallow learning curve, its widespread use, and the fact that both textbooks offer many simulation and analysis examples written in MATLAB.

Classroom

In the classroom portion of the course students use MATLAB to simulate neuronal networks as they explore topics including stimulus encoding and neural decoding, central pattern generators, auto-associative memory, tonotopic map formation, and information theory (all following the Anastasio text—see Table 1 for more detail). A typical class begins with a 10-15 minute lecture framing the topic and introducing a relevant computational model, followed by 35-40 minutes of students exploring the topic for themselves by actually running simulation code. (It is therefore essential that each student have their own computer in the classroom.) In most cases students do not write their programs from scratch, but start with a MATLAB program provided by the Anastasio textbook.

Week	Topics
1	Overview of computational neuroscience; basic computations and plotting in MATLAB
2	'for' loops and simulating random spiking in neurons
3	Relational operators, writing functions
4	Rate coding, population coding, constructing tuning curves
5	Gill-withdrawal reflex in *Aplysia*, positive feedback and leaky integration
6	Vestibulo-ocular reflex, velocity storage, central pattern generators
7	Lateral inhibition and edge detection, activity bubbles
8	Review and mid-term exam
9	Mid-semester break
10	Neuronal synchronization, epilepsy, Parkinson's disease, network theory
11	Auto-associative memory, Hopfield networks
12	Hippocampus and anterograde amnesia, distributed versus sparse representations
13	Kohonen algorithm, brain maps
14	Orientation selectivity in the visual cortex, intro to information theory
15	Information transmission in neural networks, relationship of information theory to brain maps

Table 1. Overview of topics covered in the classroom portion of the course.

For example, the introduction to auto-associative memory models starts with a short lecture on Donald Hebb's hypothesis that "neurons that fire together wire together," followed by several examples to illustrate the concepts *pattern completion* and *pattern separation*. Students then run a simulation exploring how different patterns of neural activity alter the connectivity between neurons according to Hebb's hypothesis, then use the resulting network connectivity (represented as a matrix) in another simulation to explore pattern completion and pattern separation. Students individually complete a worksheet guiding them through the exploration at their own pace (though they are encouraged to collaborate), while the instructor answers questions as they arise.

Lab

In laboratory students work individually to write code to process real-world data. Examples include constructing a plot of firing rate versus time from a raw voltage trace recorded from the Backyard Brains cockroach preparation (Dagda et al., 2013), constructing a tuning curve from raw data recorded from the macaque motor cortex (data accessed from the *Matlab for Neuroscientists* text), and generating a movie from raw calcium fluorescence data recorded from a neuronal culture (see Table 2 for more details on topics covered in lab). At the beginning of the semester, to keep students within their zone of proximal development (Chaiklin, 2003), they are provided with a "scaffold" of code to start each lab, with blank lines which they must complete in order to get the program to run successfully.

Week	Topics
1	Intro to MATLAB, Part 1 (Chapter 2, *Matlab for Neuroscientists*): Basic functions and computations
2	Intro to MATLAB, Part 2 (Chapter 2, *Matlab for Neuroscientists*): Plotting
3	Intro to MATLAB, Part 3 (Chapter 2, *Matlab for Neuroscientists*): Functions and 'while' loops
4	Rate coding cockroach lab: Use data obtained from Backyard Brains SpikerBox to plot firing rate versus time
5	Neural encoding (Chapter 13, *MfN*): Construct tuning curves of neurons from macaque motor cortex
6	Integrate-and-fire neuron model
7	Practice lab exam
8	Lab Exam
9	Mid-semester break
10	Fourier analysis I: Analyze signals to determine dominant frequency components
11	Fourier analysis II: Plot frequency of signal versus time in order to identify epileptic seizure
12	Image processing: Analyze calcium fluorescence data from neuronal culture
13	Dialog boxes: User identifies whether or not a fluorescence trace includes neural activity
14	De-noising signals: Use moving average and moving median techniques to remove noise from fluorescence signals
15	Final Lab Exam

Table 2. Overview of topics covered in the lab portion of the course.

The following is an example of such scaffolded code, taken from the cockroach lab mentioned above:

```
%% Section 1: Import and plot the data

%read in voltage trace from cockroach
experiment
[voltage, srate]= ??? ;
```

```
%create a 'time' array, for which time(ii)
corresponds to voltage(ii)
time= ??? ;

%plot the voltage as a function of time
plot(time,voltage)
xlabel('Time (sec)','fontsize',18)
ylabel('Voltage (arbitrary
units)','fontsize',18)

%% Section 2: process the waveform so that
you have a signal of just 1's and 0's: 1's
for spikes and 0's for everything else

%set threshold to separate spike events
from noise; this is somewhat arbitrary
threshold= ??? ;

%make a copy of 'voltage'
spikes=voltage;
%set every entry in 'spikes' which is less
than 'threshold' equal to 0
spikes( ??? )=0;

%in order to convert each spike into a
SINGLE 1 (rather than multiple 1's), take
the discrete difference of 'spikes'
spikes=diff(spikes);
spikes=[spikes; 0]; % add one extra entry
to 'spikes,' so that it has the same number
of elements as 'time'

%set every entry in 'spikes' that is less
than 'threshold' equal to 0
spikes( ??? )=0;
%set every entry in 'spikes' that is
greater than or equal to 'threshold' equal
to 1
spikes( ??? )=1;

figure
plot(time,spikes)
xlabel('Time (sec)')
ylim([0 1.1])
```

As the semester goes on these code scaffolds become sparser, until in the last lab students write the program entirely from scratch.

Table 2 lists the topics covered in lab throughout the semester. Topics in the first half of the course were selected with the goal of developing coding competency using neuroscience-motivated examples. Topics in the second half of the course focus on imparting the skills most universally relevant to analyzing neural signals in a graduate lab. Students with a fundamental understanding of Fourier analysis, facility with image processing techniques, and the ability to write a program in which the user interacts with the data should be capable of immediately contributing to most neuroscience labs.

In addition to teaching important data analysis techniques, the lab provides an essential environment for students to hone their programming skills. Computer programming can be very frustrating to learn (as Joseph Campbell (1988) said, "Computers are like Old Testament gods: lots of rules and no mercy"), and dedicating three hours each week to coding with an instructor nearby to answer questions can save students from hours of frustration.

Assessment

In the classroom portion of the course, students submit weekly homework assignments based on exercises from the Anastasio text. They often start these homework problems as part of in-class explorations, then finish them outside of class. They also take a midterm exam and a final exam, each with 10-15 essay questions pertaining to information processing by neural systems.

The lab portion of the course also includes midterm and final exams, each requiring students to download and analyze real-world neural data sets. It is helpful to have a practice exam during lab the week before the midterm exam (and possibly also before the final exam), so that students know what to expect.

Finally, all students have the option to conduct a final project in place of the laboratory final exam. There is large variation in how quickly students master the ability to program in MATLAB, with some capable of writing programs from scratch halfway through the semester, and others (in rare cases) who still struggle to write a *for* loop by the end of the course. A final project provides an avenue for advanced students to apply their skills to a real-world problem they find interesting. Students who choose this option consult with the instructor on an appropriate topic, which may involve computationally modeling a neural system and/or analyzing neural data (perhaps from their own research). At the end of one month of work (in which students submit updates and meet with the instructor each week), students give a 10-minute presentation and submit a paper detailing their results. Past projects have included analyzing electro-oculogram data to investigate habituation of saccadic eye movements, exploring the effects of caffeine and alcohol on neural activity in the cockroach metathoracic leg, and analyzing the network structure of the *C. elegans* connectome.

The distribution of final grades in the course was 3 A's, 1 A-, 1 B+, 1 B, and 1 C+.

RESULTS

To assess student confidence in the skills and concepts covered in the course, a questionnaire was administered on the first and last days of the Spring 2015 semester. A Likert scale was used to score the anonymous responses. Note: all seven students in the class completed the questionnaire at the beginning of the course, but only six completed the questionnaire at the end.

The statements in Fig. 1A were selected to assess students' comfort with computer code, which is foundational to Learning Objectives #2 and #3 (modeling neural system and analyzing neural data, respectively).

The data depicted in Fig. 1B demonstrate significant increases in students' confidence in their general computer programming skills.

A

	Survey statement
S1	I am capable of reading basic computer code.
S2	I am capable of writing basic computer code.
S3	If I were in a research lab, I would be capable of writing my own computer code to analyze data.

5: Strongly Agree
4: Agree
3: Neutral
2: Disagree
1: Strongly Disagree

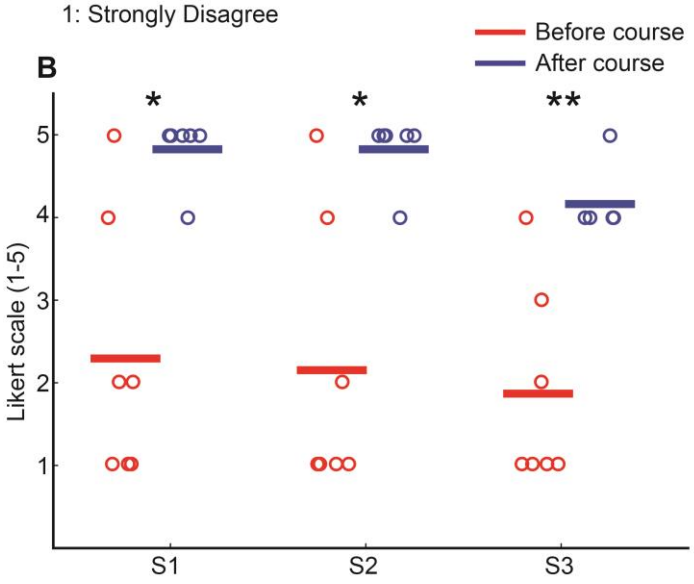

Figure 1. Students' self-reported agreement with three statements pertaining to their computer coding skills. Statements and response options shown in *A*, results shown in *B*, with horizontal lines indicating mean Likert values. The one-tailed Mann-Whitney U test was used to determine statistical significance (* = p<0.05, ** = p<0.01).

The statements in Fig. 2A were selected to obtain students' assessment of how well Learning Objective #3 was met. Students' confidence in their ability to successfully apply fundamental data analysis techniques to neural data also increased significantly, as shown in Fig. 2B.

Finally, statements 1 through 4 in Fig. 3A were selected to obtain students' assessment of how well Learning Objective #1 (understanding fundamental principles of information processing in neural systems) was met. Fig. 3B shows that students felt they improved significantly in their understanding of core principles of neural information processing. Statement 5 probed students' perception of the relevance of Computational Neuroscience to understanding the brain. The data indicate this increased only marginally, since most students started the course already thinking CN to be highly relevant to investigating the brain.

A

	Survey statement
S1	I am capable of using Matlab to transform a neuronal voltage trace into a plot of firing rate versus time.
S2	I am capable of using Matlab to construct a neuronal tuning curve from spike time data.
S3	I am capable of applying Fourier analysis to characterize the dominant frequencies of an EEG signal.
S4	I am capable of using Matlab to produce a movie of a neuronal culture's activity from raw fluorescence data.

4: Yes, without any help from my instructor
3: Yes, with some help from my instructor
2: Yes, but with a lot of help from my instructor
1: No, not at all

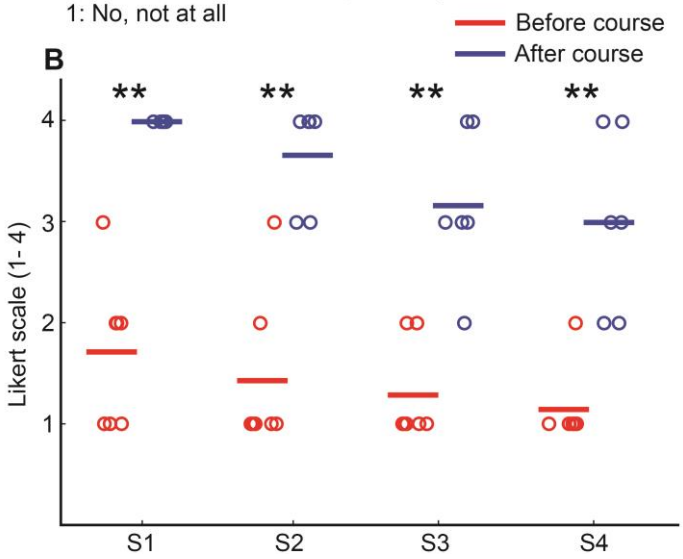

Figure 2. Students' self-assessment of skill level with four fundamental data analysis techniques. Techniques and response options shown in *A*, results shown in *B*, with horizontal lines indicating mean Likert values. The one-tailed Mann-Whitney U test was used to determine statistical significance (* = p<0.05, ** = p<0.01).

DISCUSSION

Overall, the data indicate a dramatic increase in student facility with core CN concepts as well as with data analysis techniques. Students reported significant increases in their understanding of all four assessed CN concepts (Fig. 3), although their reported understanding of information theory was noticeably lower than the other three subjects, suggesting that perhaps more time should be spent on this topic. Likewise, students appear to be slightly more comfortable with constructing firing rate plots and tuning curves than with applying Fourier analysis and producing movies in MATLAB (Fig. 2). This may be due to the fact that the latter two skills were introduced later in the semester, when the scaffolded computer code provided fewer "hints."

A

	Survey statement
S1	I understand how rate coding and temporal coding are involved in processing information in the brain.
S2	I understand how neuronal tuning curves are involved in processing information in the brain.
S3	I understand how the brain can store memories through the adjustment of synaptic weights.
S4	I understand how information theory can be applied to characterize information transmission in neural networks.
S5	Computational neuroscience offers a valuable perspective in the effort to understand brain function.

5: Strongly Agree
4: Agree
3: Neutral
2: Disagree
1: Strongly Disagree

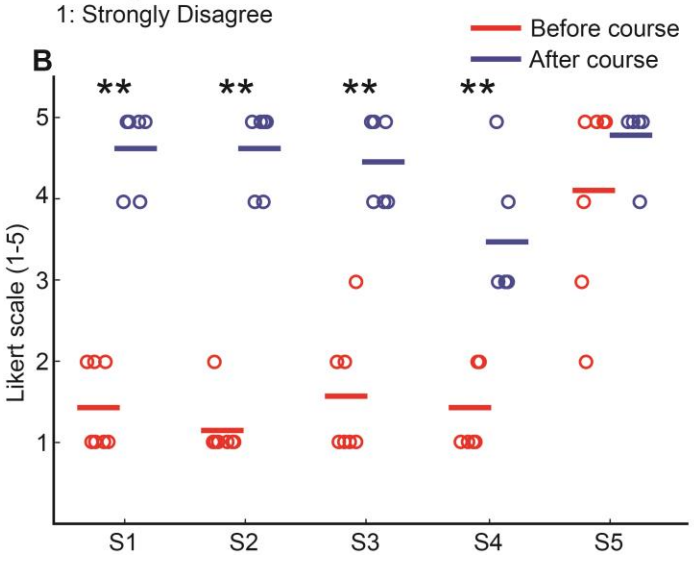

Figure 3. Students' self-reported agreement with five statements regarding their comprehension of information processing by neuronal systems. Statements and response options shown in *A*, results shown in *B*, with horizontal lines indicating mean Likert values. The one-tailed Mann-Whitney U test was used to determine statistical significance (* = $p<0.05$, ** = $p<0.01$).

A previous study has shown that just a few exposures to MATLAB-based CN laboratory exercises over the course of a semester dramatically increase student comfort interacting with computer code (Nichols, 2015). The course presented in this paper extends that study by demonstrating that students without any previous programming experience may learn to use MATLAB to effectively simulate neural systems and analyze neural data in a one-semester, algebra-based CN course.

This course has been taught four times at Ohio Wesleyan, with a total of 22 students completing it. Most have been neuroscience majors, though computer science, physics, and zoology majors have also taken it. (Most students from outside neuroscience have taken the course in order to learn basic computer programming skills in preparation for graduate studies.) The course has seen several major modifications over the years, including spending more time on fundamental programming skills at the beginning of the course, incorporating a practice midterm lab exam, and expanding the portion of the course that reviews matrix computations.

Replication of a similar course elsewhere should be straightforward, as the only major resource required is a computer lab with MATLAB software. (While Python could in principle be used in place of MATLAB, it would require translation of the computer code accompanying both textbooks.) All course materials, including a course syllabus and raw data for analysis in lab, are freely available at http://chrisfink.xyz/teaching/teaching_resources.html.

REFERENCES

Campbell J, Moyers B (1988) The Power of myth. (Flowers BS, ed) New York, NY: Doubleday.

Chaiklin S (2003) The zone of proximal development in Vygotsky's analysis of learning and instruction. In: Vygotsky's educational theory in cultural context (Kozulin A, Gindis B, Ageye V.S, Miller SM; eds), pp 39–64. Cambridge, UK: Cambridge University Press.

Dagda RK, Thalhauser RM, Dagda R, Marzullo TC, Gage GJ (2013) Using crickets to introduce neurophysiology to early undergraduate students. J Undergrad Neurosci Educ 12:A66-A74.

Dayan P, Abbott LF (2001) Theoretical neuroscience. Cambridge, MA: MIT Press.

Érdi P (2015) Teaching computational neuroscience. Cogn Neurodyn 9:479-485.

Insel TR, Landis SC, Collins FS (2013) Research priorities. The NIH BRAIN Initiative. Science 340 (6133):687-688.

Nichols DF (2015) A series of computational neuroscience labs increases comfort with MATLAB. J Undergrad Neurosci Educ 14:A74-A81.

Trappenberg TP (2009) Fundamentals of computational neuroscience. Oxford, England: Oxford University Press.

Received October 13, 2016; revised January 10, 2017; accepted January 10, 2017.

This work was supported by NIH Grant No. R01-NS094399.

Address correspondence to: Dr. Christian G. Fink, 61 S. Sandusky St., Ohio Wesleyan University, Delaware, OH 43015. Email: cgfink@owu.edu