▲ Doing Maths Algorithmically in the Age of the Computer: Introducing Octave, A Comprehensive, Open-Source Software Resource

ROGER HERZ-FISCHLER
E-MAIL: roger@herz-fischler.ca

Roger Herz-Fischler is a retired Ontario mathematics professor and former secondary school teacher. He is the author of many academic articles and books (e.g., A Mathematical History of the Golden Number*), ranging in topics from abstract probability theory to art history.*

This article has several goals:

1. To put forward my view that we should be introducing high school students to an algorithmic approach to solving certain problems where either there is no nice "formula" (in the classic sense, e.g., the Pythagorean Theorem), or where a complicated situation is better resolved by working through a sequence of more manageable steps
2. To point out the virtually complete dichotomy between the Ontario secondary school computing and mathematics curricula, and how this affects students, not only at the secondary level, but also during their further studies
3. To encourage teachers to introduce students to Octave, a relatively easy-to-learn, open-source programming language, with which students can do most, or perhaps even *all* of their assignments involving statistics, data analysis, equations, matrices, plotting, etc., and which they can *continue to use* in their post-secondary studies

To illustrate, I am going to discuss a variation of the well-known birthday problem:

When we ask k students to state their birthday, we are essentially asking them to choose a number at random from 1, 2, ..., 365. Instead of 365, we can ask them to pick an integer from 1, 2, ..., n. We want to find p, the probability that at *least two* students pick the *same* number:

$$p = 1 - \frac{n}{n} \cdot \frac{n-1}{n} \cdot \frac{n-2}{n} \cdot \ldots \cdot \frac{n-k+1}{n}$$

Before you read any further, you are invited to try and calculate, using a calculator, spreadsheet, etc., the value of p when k = 100 people, and n = 3000. Now let us see how we would do this using Octave. Being a very high-level language, the commands correspond to mathematical statements, and this makes programming quite simple. Suppose that we want to create the row vector [1, 2, ..., 100], we simply write: a = [1 : 1 : 100], where the 1 in the middle indicates that we are increasing by steps of 1. To go in descending order, as in the formula for p, we simply use a negative step, e.g., b = [100 : -1 : 1]. To find the product of the numbers 100, 99, ..., 1, we need only write **prod**(b).

Multiplication and division are indicated by "*" and "/," and when applied to a row vector, the operation will be performed on each element of the vector, e.g., we could write c = (1/100)* b. Then we could write prod(c) to obtain the product of the elements of c.

So, to do the calculations for k = 100 people and n = 3000, we simply type in the following at the Octave prompt (the semi-colon prevents displaying long vectors):

```
a = [3000 : -1 : (3000 – 100 + 1)];
b = (1/3000)*a;
c = prod(b);
p = 1 – c
```

Octave quickly responds with the answer p = 0.81148. Here we have used Octave as a powerful calculator. If we wanted to perform the same calculation for many combinations of k and n, we would create a short program with two input statements:

```
k = input('how many people?');
n = input('what is n?');
a = [n : -1 : (n – k + 1)];
b = (1/n)*a;
c = prod(b);
p = 1 – c;
disp(p)   % disp = display; % is used for comments
```

The program is simply an algorithmic transcription of the mathematics. Instead of input statements, we could just as easily have created a function of k and n. If we wanted to calculate the birthday probabilities for k = 10, 20, ..., 60, we could add a loop. This is the way computing is done in the real world: use a text editor to create a program, run your program, debug it, expand its

capabilities, store the values, and do a plot.

Above, we were given k and n and obtained the value of p. We can consider this as a *direct* problem and solution. A more complicated situation occurs because of the nature of the birthday problem. If we run the last program with $k = 30$ and $n = 365$, we find that $p = .71$. This means that if you try the birthday experiment in a class of 30, you have the large probability of .29 of looking rather silly. This leads to an *indirect* problem and solution. We now start off by fixing a level of tolerance ahead of time, for example .95, which corresponds to the "19 times out of 20" used by pollsters. Our task now is to find as *large a value of* n *as possible* so that if 30 people pick a number from 1, 2, ..., n, then the probability that at least two people pick the same number is *at least* .95. In view of the fact that the quantity n appears k times in our formula for p, finding a closed-form formula for n is probably an impossible task. However, by thinking algorithmically and then using an extension of the above Octave program, we can easily find the largest value of n for given values of k and p.

The algorithm for finding n:

1. Start with the value for k and the desired level of tolerance.
2. Begin with $n = k$ (when p is virtually equal to 1).
3. For each value of w, find p in terms of k and n.
4. Is p greater than the desired level?

   If yes, increase n by 1 and redo the calculation for p.

   If no, n is now too large; subtract 1 and stop.

To translate this algorithm into an Octave program, we take the above program and put it inside a "while-loop." To simplify, we will take $k = 30$ and set the tolerance at .95. The program would then read:

```
n = 30;   % start with n = k = 30
p = 1;    % we start above the tolerance of .95
while p > = .95
   n = n + 1;    % increment n by 1
   a = [n: – 1:(n – k + 1)];
   b = (1/n)*a;
   c = prod(b);
   p = 1 – c
endwhile    % loop again while p > = .95
disp('maximum n is'), disp(n – 1)   % we went too far,
so use n – 1
```

If we run this program, we find that if we want to be certain 19 times out of 20, then the maximum allowable value of n is only 155, which is well below 365. For 40 students, we can go up to 273, and in an auditorium with 100 people, up to 1685!

Here is another set of direct/indirect problems. The direct problem is taken—with a modification of the wording—from a Grade 12 textbook: "At a manufacturing plant, 35% of the employees call in sick on any given day. If the plant has 20 employees on the payroll, what is the probability that no more than 7 call in sick on any given day?" The corresponding indirect problem would be: "What is the *minimum* number of employees required so that the probability that *no more than* 7 employees are sick on a given day is *at least* .90?" To solve this, we would proceed in the same algorithmic manner as with the pick-a-number problem above.

Some Thoughts on Mathematics and Computer Science Curricula

If you download the 72-page Ontario Computer Science curriculum and search for "mathematics," you will note that there are zero occurrences. All that we find are vague references to "mathematical literacy." We also learn that in course ICS4U (A3.5), students will learn algorithms to perform matrix operations. In other words, there is apparently no place for applied computing—mathematics, as such, lives in another, separate world. In my opinion, Ontario mathematics education reflects the 1980s era, as far as computing is concerned. When many of our students go on to university or college, they will not likely be using spreadsheets or programmable calculators, but rather, working with real computers. Perhaps what is needed is a course entitled something like, "Applied Computing for Mathematics and Science," so that students will be prepared for the practical side of the next stage of their studies.

Why am I advocating that Octave be introduced into the Ontario secondary school curriculum? To begin with, every engineering program in Canada uses Octave (or a comparable, commercial product such as MATLAB) in their courses. Students in higher-level science courses often need to do applied programming, complicated graphing, etc. In first- and higher-year courses in linear algebra, students learn about row reduction (another example of an algorithmic process). Octave easily handles this, as well as every other linear algebra calculation.

Further, at the high school level, Octave would be ideal for the course "Mathematics of Data Management." With this one tool, students would be able to do all the

computations, and do them in a *systematic* manner. As an example, suppose we take 11 measurements on each member of a group of people. For each person, we create a 1 x 11 row vector of data. We label these row vectors a_1, a_2,.... We then create a matrix DATA1 by simply listing the vectors and separating them by semi-colons: DATA1 = [a_1 ; a_2 ; ...]. Then all we have to do is apply the Octave functions—"mean," "median," "mode," "std," "quantile," etc.—to DATA1. If later on we have another data set, DATA2, we can work with the two sets separately and then see what happens when we combine them via DATA3 = [DATA1 ; DATA2]. Simple commands also allow us to obtain bar graphs, to generate large amounts of random data, or to toss coins thousands or millions of times.

Octave

Octave originated in 1988, when early versions were used in chemical engineering courses at the University of Wisconsin. Unlike the commercial MATLAB, Octave is open-source and thus may be freely downloaded for Linux and BSD (the two systems used on most mainframes), Windows, and Mac OSX [**www.gnu.org/software/octave/download.html**].

For the benefit of teachers and students, I have put together a series of support resources that can be freely downloaded from the website: **web.ncf.ca/en493/STUDENT_LINUX/STUDENT_LINUX.html**. The resources include, in addition to software, my instructional booklets entitled *An Introduction to Octave for High School and University Students* and *An Introduction to Student Linux*. Linux, because of the possibility of having multiple "desktops," is ideal for methodical teaching and learning. The booklets can be freely copied and distributed to colleagues and students. All of the software, including the Linux distribution itself, is open source so that there are absolutely no user fees or restrictions associated with their distribution or use.

Acknowledgements

I read about the pick-a-number problem at least 40 years ago, but I have not seen it mentioned since. Unfortunately, I no longer have the reference, but as I recall, it was in a magazine for high school teachers of mathematics. A special thanks to Bruce McLaurin, the head of mathematics at Glebe Collegiate in Ottawa, for taking time to discuss the high school curriculum with me. I also much appreciated the comments of the various reviewers concerning the original version of this article.

References

Haese, R., Haese, M., Humphries, M., Haese, S., & Owen, J. (2004). *Mathematics for the international student.* [The direct factory question above was adapted from p. 723, exercise 3.]

Herz-Fischler, R. (2014). *An introduction to Octave for high school and university students.* [The text can be freely downloaded (**web.ncf.ca/en493/STUDENT_LINUX/STUDENT_LINUX.html**), may be modified, if desired, and then reproduced, on a non-commercial basis, for distribution to students.] See section 11 on how to present the solution of simultaneous equations by the method of row reduction and also by using Octave. For data analysis and statistics, see sections 05, 10, 19. Matrix operations are discussed in section 20.

Herz-Fischler, R. (2000). *The shape of the great pyramid.* [See the notes relating to the computations, p. 197. All computations were done in a systematic, algorithmic fashion; for certain repeated calculations, e.g., Pythagorean theorem, laws of sine and cosine, functions were first defined in Octave files and then called when needed.]

Ontario Ministry of Education. (2008). *The Ontario curriculum, grades 10 to 12: Computer studies, 2008 (revised)*. Retrieved from **www.edu.gov.on.ca/eng/curriculum/secondary/computer10to12_2008.txt** ▲

Encyclopedia of Mathematics Education (S. Lerman, 2014) Excerpts

"The politics of language are to some extent self-evident in the structure of the research community. In particular, English is the predominant language of this community; the leading international journals and conferences all prefer English. ...There are thousands more languages in the world that are entirely absent from mathematics education research discourse. The preference for English makes things easier for English-speaking researchers...and more challenging for everyone else. It also, however, privileges certain ways of thinking about mathematics, teaching and learning, while rendering invisible other alternatives." (p. 335)

Barwell, R. (2014). Language background in mathematics education. In S. Lerman (Ed.), *Encyclopedia of mathematics education* (pp. 331–336). London, UK: Springer.