*What kind of science is Computational Science*

by Rhett Allain

1. The author of this article, Rhett Allain, claims that popular opinion dictates that there are three parts of science. What, specifically, does Allain name each of these three parts? For each part of science that Allain identifies, recapitulate the corresponding definitions he provides?

2. What do you think is Rhett Allain’s main thesis in this article? What is he trying to say in this work?

3. List at least **TWO** of your favorite quotes from this article. Next to each quote, explain what you like about this quote and how this quote relates to your own understanding of scientific computation.

EXAMPLE: Here is one of my (Jeff’s) favorite quotes from this article:

 Quote from page 2: “At the most basic level, science is about building models. If you just build a model with no connection to real life, that isn’t science.”

Reasoning: This quote captures the essence of the scientific method applied to real-world problems. We use experimentation and observation to make conjectures about the world and develop theory that (hopefully) describes and explains the phenomena we are studying. Then, we do further experimentation to test our theories and refine our conjectures.

I love that the author is focused on describing science as a tool that is useful in the real world. This quote captures my feeling about the argument between pure and applied mathematics. As a younger man, I studied pure math quite deeply but eventually converted to applied math because I felt my work should have deep connections to real life.

*What kind of science is Computational Science? A Rebuttal.*

by Tamara Kolda

4. As described by the author, Tamara Kolda, in this article, the 2005 U.S. Presidential Committee on Computational Science provides a thorough definition of *computational science*. What is this definition? What are the three distinct elements of this definition? What is the focus of each of these three elements?

5. What do you think is Tamara Kolda’s main thesis in this article? What is she trying to say in this work?

6. List at least **TWO** of your favorite quotes from this article. Next to each quote, explain what you like about this quote and how this quote relates to your own understanding of scientific computation.

EXAMPLE: Here is one of my (Jeff’s) favorite quotes from this article:

 Quote from page 2: “Going from application area to computational results requires domain expertise, mathematical modeling, numerical analysis, algorithm development, software implementation, program execution, validation and visualization of the results. CSE involves all of this. Computer models are ultimately discrete approximations of continuous phenomena.”

Reasoning: This quote reminds me of Lloyd Trefethen’s definition of Numerical Analysis. I also love that this quote captures the essence of multidisciplinary work. The challenge and beauty of modern day scientific computation is that scientists who want (or need) to write code will do a lot better if they master a relevant set of mathematical and computational tools. One of the reasons I became a teacher is because I think our school system (K12, Undergraduate, and Graduate-level) does a very poor job of helping students learn these really important skills. I want one of my professional contributions to be a meaningful re-design of lower-division mathematics curriculum to integrate foundational ideas in CSE into our classes in a way that is meaningful for each and every student.

*Science has only two legs*

by Moshe Y. Vardi

7. What do you think is Moshe Vardi’s main thesis in this article? How does Vardi think computation relates to theory and experimentation?

8. This is the third article you’ve read that describes a debate between thinking of science as having two pillars (theory and practice) versus of thinking of science as having three pillars (theory, practice, and computation). What do you think: does science have two pillars or three pillars? Explain your reasoning.

9. List at least **TWO** of your favorite quotes from this article. Next to each quote, explain what you like about this quote and how this quote relates to your own understanding of scientific computation.

EXAMPLE: Here is one of my (Jeff’s) favorite quotes from this article:

“At the same time, *computational thinking*… thoroughlypervades both legs. Computation is the universal enabler of science, supporting both theory and experimentation. Today the two legs of science are thoroughly computational!”

Reasoning: I love thinking of computation as part of both theory and experimentation. Personally, I don’t like to think of computation as a third leg of science. Instead, I prefer to think of computation as a necessary component of every great scientists arsenal of tools. I think of computation like I think of lab equipment, or the ability to read and write. This quote captures my thinking on this quite well.

On the other hand, I recognize why we are having the debate. If we think of computation as a third pillar, our education system is much more likely to take this field seriously (which we do not do very well at the moment). Could you imagine getting a degree in college degree in chemistry or physics without ever doing work in a laboratory. Hopefully, that idea seems ludicrous. Yet, we allow tens of thousands of students to graduate from college with science degrees without ever having taken classes that specialize in scientific computation. Moreover, most introductory computer language classes focus on syntax rather than major themes of software development and computation. This approach has to change. In my opinion, we should be teaching coding in every single lower-division science course we teach in the college system. I hope to be part of a generation of educators that designs and implements such a change. My first goal is to get this done at Foothill college.

*Error: why scientific programming does not compute*

by Zeeya Merali

10. What problem(s) does Zeeya Merali describe in this article?

11. How are these problems related to your life as a student and future course work you plan to complete? Pretend that you were serious about training yourself to use scientific computation in your future career. What are some suggestions that Merali mentions that you would like to learn more about. Why do you feel you might value these suggestions.

12. List at least **TWO** of your favorite quotes from this article. Next to each quote, explain what you like about this quote and how this quote relates to your own understanding of scientific computation.

EXAMPLE: Here is one of my (Jeff’s) favorite quotes from this article:

“The level of effort and skills need to keep up aren’t in the wheelhouse of the average scientist. As a general rule, researchers do not test or document programs rigorously, and they rarely release their codes, making it almost impossible to reproduce and verify published results generally by scientific software, say computer scientists.”

Reasoning: When I read this quote between the lines, I see huge potential for my students. Researchers are some of the most well trained people on earth. Usually these people have been through 4 years of undergraduate studies plus at least 5 more years of a masters and PhD program. And yet, the ability to code well is a rare and valuable skill. Thus, if my students train themselves to code effectively, they will have an uncommon skill compared with most other researchers. Any student that takes this practice seriously can then leverage this skill to solve real-world problems and improve the practice of science at the institution of their choosing. This quote makes me want to work harder as a teacher to empower my students!

*Best Practices for Scientific Computing*

by Greg Wilson et. al.

13. What problem(s) do the authors describe in this article?

14. What recommendations do that authors make in order to address the problems they describe?

15. List at least **TWO** of your favorite Best Practices that are listed in this article. Next to each Best Practice that you choose, explain what you like about this practice and why you feel it’s important. Then write a few sentence describing how you might implement this practice in our ENGR 11 course.

EXAMPLE: Here is one of my (Jeff’s) favorite Best Practices from this article:

“Make incremental changes: Work in small steps with frequent feedback and course correction.”

Reasoning: I believe that this is the best way to approach large problems. I live my professional life with this practice in mind. I know that great accomplishments come after long sequences of productive struggle on hard problems. I also know that the best way to build is by working slowly and steadily, making errors along the way. This practice taps into my knowledge and understanding about how learning works.

Aside from that general discussion, I also know that software development is challenging and takes a lot of time. When deep into a project, small errors can be detrimental and in fact fatal. I have spent over 24 hours of working time debugging a program that was missing a single right parenthesis symbol. I had forgotten to type that symbol and did not run the entire project when updating the few lines that I was working on. Then, a few days later, I re-ran the code and it didn’t work. I spent lots of time trouble shooting and trying to figure out what the hell had happened. After almost a week, I discovered that the problem was not in my design, approach, algorithm, or thinking. It was with my syntax. Had I re-run my code after that small correction, I would have probably caught this immediately. But, the fact that I waited and did multiple updates over a period of days before I ran the entire project had horrible effects on my levels of stress and productivity. When I read this best practice, I feel that pain all over and wish I could have communicated this practice to my younger self… I would have avoided so much pain over such a simple mistake.

In terms of how I implement this now, I have multiple habits I use. First, I try to live by the rule that every time I write an open parenthesis, I immediately type the close parenthesis. Second, I often design small test problems for larger projects. One of the issues that came up with the project I mentioned above is that the run time on that project was quite high (on the order of many hours). I was in a pinch and wanted to avoid waiting to have to check simple updates. But that ended up costing me many hours of debugging time. One habit I’ve adapted since then is to come up with representative examples to test my code that capture the essential features of the problem but also allow for quick iterations and feedback. That habit costs a lot of time up front since generating such examples is nontrivial. But over the life of the project, I reap the rewards because I can iterate quickly. I have many other habits I’ve adapted to address this issue but that is a good start for this reflection.

16. After reading this article, think about why you are in this class. Why do you think this course is required for some STEM degrees? Make specific references to the articles you’ve read. What problems are described in these five articles that relate to the decision to require MATLAB?

17. How might your work in this course benefit you in the future? How might your learning help you avoid some of the problems described in these articles? Make specific references to some of the articles as you think about how this course relates to your future learning?

18. I guarantee that there is no way you are going to learn everything about MATLAB that you might need in your future classes. In other words, in order to really take advantage of this course, I believe you will need to make an explicit effort to build MATLAB learning into your future academic plan and find opportunities to learn MATLAB beyond ENGR 11.

What benefits might you enjoy if you make this effort? Specifically reference the articles as you play out a thought experiment in which you make a concerted effort to learn MATLAB beyond this course. How might you take advantage of this effort in your near and distant futures.