

## 2011 DOE CSGF Essay Contest 'Emerging Writer' Winners



*The DOE CSGF annual essay contest engages fellows and alumni with the opportunity to write a popular science composition on a topic of personal importance. The authors of the 2011 essays featured below were recognized as "Emerging Writers" and worked with a professional science writer to improve the compositions and further enhance their writing skills. We hope you enjoy the results.*

---

### A Pencil Sketch of Computing Progress



**by Joshua Hykes**  
*DOE CSGF Alumnus*

In 1958, Leonard Read described – in agonizing detail – how a pencil is made. His essay, entitled *I, Pencil*, begins in the cedar forests of California and Oregon, where trees are harvested with saws, ropes, and trucks. In a mill, the logs are cut into slats and kiln dried. In the pencil factory, the slats are shaped into pencil halves, leaving room for the lead.

Read also relates the manufacture of the lead, lacquer, metal cap and eraser. While the details become boring, the cumulative effect is convincing: No one person could ever make a pencil. Read's purpose is to demonstrate the effective operation of the "invisible hand" in a free market. However, there are other lessons to learn.

Consider this: The pencil factory has it easy. Most of the labor is already done. Workers just put it all together. Obviously, pencils aren't the only useful product made by combining materials, and the process isn't limited just to physical goods. It's also true for ideas, which always build on one another. Just as the wood, graphite, lacquer, metal and eraser are assembled into a product more useful together than any of the parts alone, seemingly unrelated ideas can be pieced together to produce almost miraculous results.

Allow me to describe one area in which this is particularly evident: the use of computers to simulate the world around us. Today, for example, businesses and governments use simulation to design airplanes and electrical power plants and to predict changes in the climate and financial markets. The developers and users of these simulations are bright and hard working but, just like at the pencil factory, they get a lot of help from others.

Take, for instance, Isaac Newton and Gottfried Leibniz. More than three centuries ago these mathematicians developed a method to compute the tangent to a curve and the area under a curve. The system came to be known as *calculus*, and it is taught in schools and colleges for good reason. (If Newton's name for the concept had stuck, we would be taking classes on "fluxions.") In addition to its usefulness in geometry, calculus often is used to describe changes or flows. Mathematicians call these descriptions *differential equations*. For instance, water flowing through a pipe can be described with differential equations, as can the changes in a rabbit population. Today, most simulations use customized differential equations to accurately portray reality.

While the discovery of calculus would be laurels enough for most to rest on, Leibniz kept busy. He was famous for designing a calculator he called the "reckoner": a mechanical device using a stepped drum and a series of gears. The tools he designed were widely heralded and kept in the aristocracy's palaces and drawing rooms, but ended up being mostly for show, since they were too unreliable for day-to-day use.

This brings us to Charles Babbage, some 150 years after Leibniz. Babbage was a man of tables. In the midst of the Industrial Revolution, when science and engineering, building and shipping were skyrocketing, numerical tables met the need for arithmetic computations. Babbage was a mathematician and engineer who understood the methods used to compute the tables, as well as the practical need for them; he owned 300 volumes. Unfortunately, every number in every table had to be worked out by hand. After a grueling session of double-checking results, Babbage recalled saying, "I wish to God these calculations had been executed by steam."

Instead of just wishing, he started work on what he called his "Difference Engine," a large mechanical calculator that would produce tables automatically. He even designed an automatic printer to eliminate typesetting errors. Babbage was able to finance construction, but delays and cost overruns left the machine unfinished.

Using insights from his first calculator, Babbage took on an even bigger problem: designing a multipurpose machine that could compute more than just one table. He called his design the "Analytic Engine." Instead of needing a new machine for each task, the multipurpose machine could be reprogrammed. The idea of a programmable machine is essential to modern-day simulation. While Babbage's ideas were groundbreaking, the technology of his day left him unable to see their full potential.

Claude Shannon did his part to bridge the gap from Babbage to the modern computer. Born in Michigan during World War I, he once used two existing barbed-wire fences to build a homemade telegraph to a friend's house half a mile away. But he did much more than tinker. At the Massachusetts Institute of Technology in 1936, he used the best computing machine of the time to solve differential equations. However, it used analog methods, the technological equivalent to using vinyl discs and cassette tapes for sound recording. In what has been called "one of the most important master's theses ever written," he proposed the rules for digital

circuits, which treat everything as either “on” or “off,” 1 or 0. In the music recording analogy, this was like proposing the digital compact disc. Besides his work in digital logic, Shannon's most important idea was “information theory,” the foundation for all storage and communication of data. For example, watching movies over the Internet or from a DVD are results of information theory. Needless to say, without Claude Shannon's ideas, simulation on modern computers would be impossible.

Thus, without the ideas of Newton, Leibniz, Babbage and Shannon, trying to simulate our world on a computer would be like trying to make a pencil with no wood, graphite, lacquer or metal. By piecing their concepts together, we can make unheard-of discoveries and predictions. Today it's possible for engineers to include computer simulation as a key part of designing their newest gadgets.

## References

Leonard Read, *I, Pencil*, The Freeman, December 1958.

Jason Bardi, *The Calculus Wars: Newton, Leibniz, and the Greatest Mathematical Clash of All Time*, Basic Books, 2006.

Doron Swade, *The Difference Engine: Charles Babbage and the Quest to Build the First Computer*, Penguin, 2002.

*Claude Shannon: Collected Papers*, edited by Neil Sloane and Aaron Wyner, IEEE Press, 1993.

---

## Untitled Essay



by Paul Loriaux

*DOE CSGF Alumnus*

In Siddhartha Mukherjee's epic history, *The Emperor of all Maladies*, we learn the bitter story of 2-year-old Robert Sandler, who in 1947 found himself in the musty basement laboratory of Dr. Sidney Farber with his spleen the size of a football. Robert, the son of a Boston shipyard worker, had acute lymphocytic leukemia (ALL); his bone marrow was pumping his young body full of undeveloped white blood cells.

In one respect, Robert was lucky: His visit came on the heels of Farber's disastrous attempt to treat ALL by administering the vitamin folate. After discovering that folate only exacerbated his patients' condition, Farber treated Robert with “antifolate” instead. For a few months the following spring, Robert regained his cherubic youth and an appetite to match. Summer, unfortunately, brought with it a relapse of the leukemia that quickly took his life.

From this historic episode we can derive the two great tenets of modern chemotherapy: First, cancers can be treated chemically, but second, remission often is only temporary. Two decades

after Robert's death, researchers would attribute this phenomenon to "fractional killing": A particular dose of a particular drug will kill the same percentage, but not the same number, of target cells. So while a particularly potent drug may be 99.9999 percent effective, when faced with a tumor of 1 trillion cells, fully 1 million will survive treatment – more than enough to repopulate the tumor.

At first blush this might seem like a cruel consequence of the law of large numbers: After a multitude of identical trials we can expect to observe the average result. But herein lies the rub: Those 1 million survivors aren't identical.

One hallmark of cancer is its stubborn recalcitrance to anti-proliferation cues. Cancer cells divide with reckless abandon and their genomes consequently become unstable. After several generations the result is a heterogeneous mass of cells whose only shared trait is uncontrolled proliferation. It is precisely this heterogeneity that makes tumors resistant to chemical treatment, and it is in this theater of heterogeneity that computational science can make its greatest impact.

The molecular mechanisms by which cells respond to chemical perturbations have been studied for decades. For example, cells absorb antifolate via a chemical carrier. Once inside the cell a chain reaction blocks production of thymidine, one of the four nucleic acids required to make DNA. Faced with a shortage of this building block, the rapidly dividing cells are arrested and die off.

These mechanisms are so well studied that we can describe them mathematically. Once described, or "modeled," we can feed the model into a computer and calculate the abundance of every molecule before, during, and after chemical perturbation, in effect predicting whether a particular treatment will yield the desired response.

However, cancer's heterogeneous nature means it's not enough to simply predict how a particular cell responds to different perturbations. We must predict how *different* cells respond to the *same* perturbation. Mathematically, a different cell means a different "steady state," i.e., a state in which all reactions are perfectly balanced and no molecule is being consumed faster than it's produced nor produced faster than it's being consumed.

But finding steady states can be a real pain, the source of which is succinctly given by physicist Julian Barbour: "Nothing in nature is linear." Because chemical reactions frequently depend on two or more molecules, their rates are said to be "nonlinear." The problem with nonlinear reaction rates is that it can be impossibly difficult to find a combination of rates whereby no molecule is being produced faster than it is being consumed.

Or would be nearly impossible, but for a little trick. Chemical reaction rates are proportional to the abundances of their reactants. That is, they are *equal* to the abundances of their reactants, multiplied by some proportionality constant. This constant is often called a rate constant. It turns out that our reaction rates are perfectly linear with respect to their rate constants. If we

simply change which of these we call variable and which we call constant, we can find a linear version of our model and calculate its steady state easily on a computer. Assigning numerical values to these constants now yields a model of a particular cell and by changing the values we can model any number of different cells.

Now for the payoff. Thanks to high-performance computing (HPC), we can simulate the response of millions of different cells to any chemical treatment of our choosing. These simulations not only tell us what percentage of cells will respond to the treatment, but what *features* of the cells we should measure to *predict* the response. If enough models are made and simulated in this way, we can begin to tailor treatment of a tumor to its unique chemical composition. Gone, hopefully, will be the days of treating cancer with non-specific and highly toxic cell cycle inhibitors.

In summary, as biological scientists uncover increasingly detailed mechanisms of how cancer cells respond to different treatments, we can harness the predictive power of mathematics and HPC to tailor the therapy of each individual tumor. In our collective war on cancer, computers may end up being our most valuable allies.

- - -

## Bit by Bit, Note for Note



**by Douglas Mason**  
*DOE CSGF Fellow*

When Bob Dylan was asked how The Beatles influenced him, he said, “They were doing things nobody was doing. Their chords were outrageous, just outrageous, and their harmonies made it all valid.”

But were their chords so outrageous? What harmonies were they using that shocked him? And did they really come out of nowhere? Michael Cuthbert and Chris Ariza, two scientists at the Massachusetts Institute of Technology (MIT), are writing a sprawling software library they hope could one day answer these questions. They could learn not only what made The Beatles so “outrageous,” but also whether they have more in common with Bach or Lady Gaga – and it doesn’t stop there. The MIT team’s software, combined with fierce computational prowess and an ever-expanding database of encoded music, could address questions that long have lingered on the tongues of musicians and music industry professionals: What song will be a hit? Who is the target audience? What melodies could best encourage shoppers at the checkout counter?

I met Mike, as he prefers to be called, in a dimly lit bar at the edge of Harvard University. He had taken the subway from his office just down the street and was fidgeting with his napkin, scribbling notes that looked like they came from another planet. He explained that they were ideas for his software library.

During our conversation, Mike cited a study showing that the average American listens to 4½ hours of music every day. He didn't need to convince me: The radio blaring overhead was testament enough. Yet, as Mike pointed out, music remains one of the most poorly understood media available to us.

Regardless, several companies already say they can use computers to predict our musical preferences – and even the next chart-topping hit. Such software has permeated the culture in tools like iTunes' Genius playlist and Pandora's recommendation program. These services hire hundreds of people to document by hand features of the music they hear, such as instrumentation, genre, and voice type.

Why so much labor? While a simple question like "What note is playing?" can be trivial for a trained musician to answer, extracting that information from a recording is beyond the capacity of even the most advanced computer programs available today. As Mike put it, "Asking a computer to read in an mp3 and tell you what chord is playing is like asking a computer to read a video on YouTube and tell you if the man in it is truly in love. We just aren't there yet."

Instead, Mike's goal is to write software that is more than a database of musical features. He hopes to write a full suite of tools that can manipulate music at the most fundamental level. While music notation – the kind you read on a piano score – can tell us how to perform a piece, it is still fraught with ambiguities that Mike hopes to overcome. As a result, his software manipulates abstract objects and hierarchies to correctly represent music, and its many expressions, for the computer as well as the musician.

Of course, Mike isn't the first person to attempt this formidable task. Musicology software experienced a "golden age" in the early 1990s with the advent of Humdrum, a suite of music-analysis tools that handled the challenges of music notation by dividing each element – key changes, chord changes, and notes – into pieces strung on a line like words in text. Students learned quickly, however, that performing anything but the simplest tasks required methods so convoluted it would take years to learn them, and there were no clear ways to speed up the software's processes.

Disappointed in the state of the field, Mike was motivated to create a library of code that could do for music what Facebook had done for social networking and Google had done for online searching. It was important to him that the software be modular, easy to use, and able to capitalize on a growing body of users – all the ingredients necessary to forge the kinds of musical connections he envisioned.

So Mike, as a graduate student at Harvard University, began building on a popular scripting language called Python. Its benefits were immediately obvious: with a huge user base, Python is scalable to perform massively parallel computations on enormous datasets. Even better, its language is malleable enough to make software code readable for the educated musician – precisely the user base Mike was coveting. The challenge, however, was to secure funding, and computational musicology was difficult to sell. To get the job done, Mike hired Chris, then a

young music professor who had made a name writing software that could generate complex melodies from simple inputs. Together they applied to every agency they could find.

And it worked. Mike has since been hired and promoted to assistant professor at MIT, where he enjoys a prestigious grant from the Seaver Institute, which normally only awards money to hard science projects. He and Chris now employ a cadre of undergraduate and graduate students, plugging away to resolve the sorts of minutiae musicians normally take for granted, like how much time to allot a grace note.

Their ambitions are as large as the databases they are compiling. Most recently they have worked to apply music21, their computational toolkit, to a repertoire of more than 100,000 pieces of music from the Middle Ages to yesterday. Why such a huge collection? “Because experts’ time is limited, they’ve focused on the extraordinary,” Mike told me. “But we know almost nothing of the norms of everyday music. Just as we know more about Mozart and Haydn than all late 18<sup>th</sup>-century music, we know more about The Beatles than all of 1960s music combined.”

As Mike and Chris continue to optimize their software to scale it to high-performance computing, they have already hit a few milestones. Musicologists recently used the programs on a reservoir of traditional Chinese music and found statistical melodic differences between songs played in coastal regions and those played farther inland – providing hints to a musical genealogy in Asia.

A year ago, such a study would have been nearly impossible, and it delights Mike to see his software used for work with such large cultural implications. With their latest work, Mike hopes Music21’s reach can go even further. In fact, they hope to make history. “For so many applications of computational musicology people have been waiting a hundred years to get an answer,” Mike told me. Now there’s no reason they should wait any longer.

---

## Studying the Cosmos, One Universe at a Time



**by Paul Sutter**

*DOE CSGF Alumnus*

It was on one of those seemingly mandated cross-country family camping trips that I first saw the full night sky, unadulterated by busy city lights. Lying on my sleeping bag, staring up at that Montana big sky with my head full of wonders, I annoyed my tired parents with question after eager question: What’s the name of that constellation? Why are those stars so colorful? What makes the Milky Way so ... milky?

Many years later, I'm still asking questions about the night sky, but now I use some of the world's most powerful supercomputers to help answer them.

Why supercomputers? The problem is that the universe we live in is dark. Very dark. And not dark as in "not very full of bright things," but dark as in "filled with stuff that doesn't interact with light." Naturally, we call this stuff "dark matter," although "dark" is a bit of misnomer – implying that there's some oozing menace worming its way amongst the stars. It's more "invisible" matter. Dark matter and light are like bitter ex-lovers: When they cross paths on the street, there is no interaction, only downward stares and hurried shuffling of feet.

Fortunately, dark matter makes its presence felt through the gravity it exerts, and we can infer its existence through careful observations of how stars and galaxies move.

When we try to understand the universe through simple telescope observation, we're stuck with the unfortunate fact we simply don't see most of it. We see all the hot glowing parts: stars, galaxies, and gas clouds. The only way to understand the dark matter that comprises most of the universe is through simulation. Using high-performance computers, we can follow both dark matter and regular, visible matter as they evolve through billions of years of cosmic time. We can then make simulated observations – using simulated telescopes in our simulated universes – and compare them to the real world. By simulating various models of structure growth in the universe over eons we can make predictions and test theories – in other words, do science.

Our simulations follow the evolution of dark matter, hot gases, magnetic fields, and sometimes even gigantic black holes. We produce simulated observations in visible, X-ray and radio wavelengths. We generate relations between properties we can observe directly and the underlying dark matter structure. Finally, we track the formation and evolution of dark matter structure so we can infer histories based on present-day observations.

Performing these simulations is no easy task. I have spent a large portion of my graduate career developing the tools necessary to run, manage and analyze them on computers made of tens of thousands and even hundreds of thousands of processors.

We need an army of processors because we simulate huge portions of the entire universe for enormous spans of time. We portray such extreme volumes for a couple of reasons. First, the objects we like to study are themselves monstrously big. Second, without simulating a large volume we can't know whether what we're seeing is typical or a special case. By modeling a large chunk of the universe in one go we can calculate average structures or phenomena.

The sizes involved are almost incomprehensible: A typical cosmological simulation will portray a section of the universe one billion light-years on a side. But even using the world's most powerful computers, the smallest objects such a simulation can depict in detail will still be the size of the Milky Way galaxy.

The time spans we model are equally mind-boggling. If you think of our simulations as movies comprised of individual frames, the picture right before the one representing the present day would have been taken sometime in the late Jurassic. Only by calculating conditions at such huge intervals can we hope to portray the universe as it evolves over eons.

What benefits does society “collect” from these simulations? Certainly not better color television or advanced medicines. While cosmological simulations offer some serendipitous benefits, such as aiding nuclear fusion research with a better understanding of hot gases in extreme environments, the most important benefits are subtle yet profound. We can now begin to answer essential questions about the universe. We can directly probe its fundamental structure by lifting the veil of dark matter. We can understand the evolution of galaxies, stars, and even planets. We can even begin to predict the future of our universe: its ultimate fate and composition.

When children of the next generation are lying on their sleeping bags, staring up at the night sky, some of their questions will have answers. I, for one, am proud that my work with the world’s most powerful computers will help satisfy their curiosity and help their parents to get more sleep!