Colbyn Wadman

Dr. James R. Birrell

ENGL 2010

March 22, 2021

A Critique on the Future of Medicine

Introduction

In his video essay, A Future of Medicine, Dr. Bernard postulates on a world that views our current system of evidence based medicine, as we see European plague doctors. Because in our modern system of evidence based medicine, we extrapolate population data to a given person, for the purpose of predicting outcome. Which you may ask, what is the problem with such?

To explain, from a practical standpoint, consider this. What if, instead of running a trial on thousands of disperse persons. What if you were simply cloned, hundreds of thousands of times, and therein, we performed hundreds of thousands of trials, forming a stochastic model that likewise predicts outcome for some given treatment?

As Dr. Bernard points out, if a trial was done on a bunch of men, there have been documented cases where women taking the very same medication experienced different reactions. Nowadays, trials are required to include notable age, gender, and race/ethnicity demographics. But as Kravitz pointed out, "[this] may do nothing but ensure that the estimates for any one subgroup are unreliable due to small numbers". In the very same paper, Kravitz coined this phenomena, the "Heterogeneity of Treatment Effects".

Furthermore, Dr. Bernard likewise points out that, if this were possible, we could likewise extend this to drug development. Perhaps as I'd imagine, in a manner that may be inconceivable to us, as I will explain.

But you may ask, how exactly do we run hundreds of thousands of trials of my identical clones? We run computer simulations, as Dr. Bernard proposed. Although what Dr. Bernard proposed, was

essentially a thought experiment, explicitly regarding this as something that probably won't be viable in our lifetime. Conversely, the following paper will say otherwise.

Realization

This all began from a TedX talk by a man called Rahul Sarpeshkar (Professor of Engineering, Thomas E. Kurtz Professor, Professor of Microbiology & Immunology, Professor of Physics, Professor of Molecular & Systems Biology). In his talk, Sarpeshkar discussed the prospects of analog computing for solving differential equations in a manner that far outpaces the capabilities of digital computers, and furthermore, he claimed that we could viably simulate an entire person if we built an analog computer that spanned the size of the given auditorium.

This is due, not to processing performance, but as Sarpeshkar argues, in information theory.

That is, digital computation in contrast, while built upon analog mediums, forgoes most of the real estate therein for a limited set of gates defined in terms of a mere bit, in a multi bit analog channel. Conversely, in allowing full utilization of the unclaimed and unexploited real estate therefrom, in further expanding our conception of computation to more than mere logic, new applications may potentially become commercially viable. Such as perhaps, simulating the molecular interactions within cells, to entire tissues, to entire organ systems, to perhaps, entire persons.

Or rather, as Sarpeshkar summarized:

> "[at] low informational precision, logic basis functions simply cannot compete with the richer basis functions of analog computation that can process all the bits at once in parallel and just automatically solve the task, e.g. by using Kirchoff's current law for addition or chemical binding for multiplication."(Sarpeshkar)

Furthermore, to further expand upon the premise outlined in A Future of Medicine. I don't think this will be the full story to the totality of medicine. Because, simulations simply do one thing, they simulate, and therein, we build a probable stochastic model from such. Put simply, it's a mechanism that may be used to answer yes or no questions, but of course, what precedes treatment to some illness is the

detection of such.

In his book called “The Body: A Guide for Occupants”, Bill Bryson regarded cancer and other such diseases as “system failures”, and therein, he remarked, part of the problem is that our nervous system, paradoxically, may not register the early formation of cancer and other such events (given it’s severity). But, perhaps we can go even further than the mere detection of cancer.

What precedes treatment is detection, and when this is continuous, we may regard such as monitoring, and this is where we enter into the domain of molecular programming. Can we run computational devices within cells? Yes. Furthermore, as Sarpeshkar noted in his paper, Analog synthetic biology, electronics (simulation) and chemistry (e.g. biochemical computation) are deeply linked. As Sarpeshkar wrote,

> “There are striking similarities between chemical-reaction dynamics (figure 3a) and electronic current flow in the subthreshold regime of transistor operation (figure 3b): electron concentration at the source is analogous to reactant concentration; electron concentration at the drain is analogous to product concentration; forward and reverse current flows in the transistor are analogous to forward and reverse reaction rates in a chemical reaction; the forward and reverse currents in a transistor are exponential in voltage differences at its terminals analogous to reaction rates being exponential in the free-energy differences in a chemical reaction; increases in gate voltage lower energy barriers in a transistor increasing current flow analogous to the effects of enzymes or catalysts in chemical reactions that increase reaction rates; and the stochastics of the Poisson shot noise in subthreshold transistors are analogous to the stochastics of molecular shot noise in reactions. [...] The logarithmic dependence of the electrochemical potential in chemical concentration or of current enables one to map log-domain analog transistor circuit motifs in electronics to log-domain analog molecular circuit motifs in cells and vice versa.”(Sarpeshkar)

Therefore it appears to me that we have a sort of isomorphism between simulation on electronic infrastructure and biochemical reactions. Which I argue, has far reaching implications. Because this reduces modeling to programming, and from programming, abstraction. Why does this matter? When we abstract, we may sometimes reduce ‘complex’ things into simpler, more conceptually ‘discrete’ things.

Which therein, due to this simpler nature, may further permit others to effectively build upon such, and therein, may create something more sophisticated, perhaps even, greater than the sum of it's components.

That is, a system where discrete units build upon other discrete units and therein produce more complex non-discrete units. This system permits for abstraction, so complex non-discrete units may be abstracted into simple discrete units. Thereafter this process of production and abstraction enables further production and abstraction and so forth. Each iteration or generation may be considered to be more sophisticated than prior generations, given that each generation is a product of prior generations… From this analogy, you can imagine these bottom-up and cumulative processes will eventually give rise to very sophisticated products, and perhaps one day, akin to how emergence gives rise to the complexity found in nature.

From personal experience, my https://imager.io project wouldn't be possible without the various open source components it's built upon. Simply because my time is finite, and especially because lower-level encoding details are just too complicated for me to understand and implement on my own. I am nevertheless able to compose such components into a larger and more sophisticated end product, from the preexisting output of resources and information from the global open source, software community. Overall added value that may be considered to be greater than the sum of its components, and therefore emergent in a manner of speaking. In an old English paper I likened the open source community as "the printing press of computable knowledge", and perhaps even more significant than the advent of the printing press itself, because as the industrial revolution introduced a force multiplier of human muscle, so too does abstraction introduce a force multiplier of the human mind.

Because, as I've written, while a book may describe a life's work in mathematics and applications therein, the medium is itself rather passive. A book may describe a life's work in applied mathematics, yet a mind is required to manifest its application. Whereas, imagine a medium where the most knowledgeable of experts can record their understanding of a given domain as functions that map problems to solutions, in a manner that can be utilized by any layperson, and thereafter this record can be reapplied, reused, and so forth, forever thereafter, and, in this manner, what are the ramifications of such?

Analog Computers

Digital computation is discreet, and proceeds in terms of discrete steps. All representations in digital form, are mere, and meaningless symbols. Computer arithmetic for instance, is simply the manipulation of such symbols is manner that implements such operations. This is also, the greatest strength to discrete computation. Remember those pen n' paper procedures you were taught for adding arbitrary numbers together? Digital computation implements the very same processes, and so, can scale to adding arbitrary numbers without loss of precision(Hehner).

In this manner, the versatility of discrete computer architectures cannot be understated. For instance, is there utility in simply simulating analog computation on digital hardware? Furthermore, if the criticism is about the von neumann bottleneck, then such really pertains to the limitations of the von neumann computer architecture, and therefore doesn't disqualify the entire field of discrete computer architectures.

Conversely, the central issue plaguing analog computers is that of precision. Analog computations are typically bounded to just three or four bits of precision. Yet, advocates of analog computers argue that many problems do not exceed such limitations, and therefore permits implementation in an analog environment that affords a greater degree of optimizations that aren't possible on digital computer architectures. In a similar manner, MacLennan adds that analog environments are a natural fit for noisy and low resolution real world measurements.(MacLennan)

Likewise, Sarpeshkar argues that the fundamental limitations of analog computing, notably, noisy signals, is ideal for simulating stochastic process. Because simulating randomness on digital hardware imposes synchronization constraints that significantly impacts performance on such systems. Whereas on analog architectures, Sarpeshkar argues, you essentially get such for free.

Therefore, overall, while discrete computer architectures are incredibly versatile, such may not necessarily be efficient. Furthermore when it comes to solving, say, differential equations, analog computing is unrivaled. Consider, for instance, the following digram that implements $\ddot{y} = -y$.

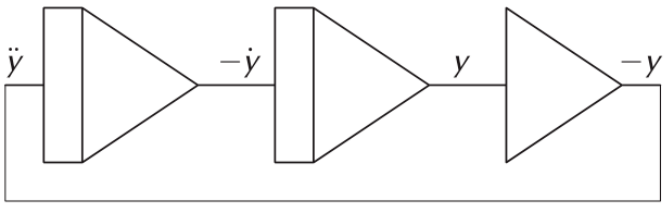

Figure 1: Source: https://chalkdustmagazine.com/features/analogue-computing-fun-differential-equations/. (Note that this is missing initial conditions.)

In the above figure, the leftmost elements are the integrators, while the rightmost element is called the summer, and each elements implicitly flips the sign. Personally, there is an elegant economy to the above diagram, and furthermore, it implements such without requiring stored computer memory, and this itself is notable (as I will try to explain shortly). In contrast, implementing the same processes on digital computer architectures would require significantly more infrastructure in implementing the same mathematical laws in terms of manipulations of arbitrary symbols. Which obviously increases overall costs, size, energy requirements, and so forth.

As MacLennan wrote, in digital computation, quantities are rather arbitrary symbols that have no direct relationship to the physical systems that such may be tasked with simulating. In contrast, generally speaking, analog and physical phenomena are governed and defined by the same mathematical laws. Or as MacLennan put it, "the computational quantities are proportional to the modeled quantities."

With regards to latency. Imagine we somehow needed to compute one instruction for each atom, for each cell, in the human body. Which would then amount to $4\text{x}10^{27}$ instructions.

To imagine this in terms of time. Consider first, simply the latency overhead incurred when your processing hardware and main memory is physically separated from each other, and therefore, each instruction must first read some datum from main memory. Lets say this overhead is 100ns per instruction. Then latency overhead alone will amount to $4 \text{ x } 10^{20}$ seconds, or in other words, latency overhead alone will total $12,683,916,800,000$ years per evolution.

But alas, this is a ludicrous exercise for a multitude of reasons. Because, for instance, what is meant by one instruction per atom? For each evolution, can the state of each atom be computed without

considering neighboring interactions? But crucially, remember the aforementioned digram that implements $\ddot{y} = -y$? Such problems needn't concern such implementations.

Business Model

The aforementioned diagram isn't ideal, on the basis that the program is essentially hardcoded or defined in the hardware configuration. Ideally, to suit the varying needs of industry, what we need are more general purpose analog computer devices. That is, they need to be programmable.

From personal experience, there seems to be costs associated with generality, and presumably, the same extrapolates to analog devices.

Likewise, from personal experience, it seems as though technical innovation itself occurs in an incremental fashion. As though there is never some removable jump discontinuity in the graph of progress. On this grounds, working backwards, before we see whole person simulations, we will perhaps see simulations of organ groups, to individual organs, to individual cells, to individual molecular interactions within a given cell.

In the same manner, how do we define the given system that we intend to simulate? Do we begin in a bottom up fashion, and so derive quantum mechanical models of our system? Or do we begin with higher level definitions of such? In the latter, abstracting whole swaths of agents into simpler and more concise definitions, may reduce resource expenditure, but at the expense of accuracy. Presumably there exists some mean between the two tradeoffs.

For example, Sarpeshkar defined the computational environment of cells as consisting of "highly computationally intensive nonlinear, stochastic, differential equations with 30000 gene–protein state variables that interact via complex feedback loops." If our concern pertains to such a system, a top down solution would begin with a more direct model of such. While a bottom up solution would begin with the quantum mechanical physics governing such a system, which may be unnecessarily (if such is viable).

From a general perspective, we may presume that before we see computationally expensive "bottom up" simulations. Some mixture between the two tradeoffs will manifest, but learning more towards higher level, top down definitions of such.

Furthermore, before such evolutions may occur, each phase that manifests in the evolution of our simulation capabilities must be commercially viable, and to seed such, the initial phase in particular must to be commercially viable.

Therefore, starting from the beginning. If simulating small molecules and biochemical interactions is the simplest of such evolutions. Then perhaps it is best to begin with a business model founded upon offering solutions to the computational chemistry and biology industry. Which, according to "Grand View Research, Inc", is expected to reach $13.6 billion, by 2026.(Grand View Research, Inc)

But, we should constrain the "Minimum Viable Product" (MVP) to computational chemistry, presuming such to be the simplest overall strategy. In this regard, the utility therein is multitude. But notably at the time being: run experiments without having to first figure out how to physically synthesize the given molecules in a lab, and more generality, cheaply iterate through numerous candidates prior to cost intensive research.

With the industry established, we may forgo full generality in our analog computer architectures, in the interest of application specific optimizations. Ideally, this also significantly simplifies supporting infrastructure. For instance, we needn't "figure out" how to implement general purpose analog software and programming languages.

Although a foreseeable disadvantage of an application specific software ecosystem will perhaps pertain to a smaller experienced labor pool. Furthermore, generality entails wider applicability, and so therefore, costs will likewise be further spread out from a larger consumer pool. But at the same time, if personal programming experience is applicable, it is very difficult to engineer generality in ad-hoc systems when the interactions therein isn't obvious. But very easy to do so after such systems manifests. Because we can see the interconnections and repetitions therein and -at the very least- abstract such into fewer reusable components. Therefore, if our initial viable product is successful in industry, practical generality will naturally follow.

Furthermore, specialized tools may lead to simpler tools, perhaps akin to the Elm language and it's more constrained frontend framework therein.

From a business perspective, we want our customers to be deeply integrated with our products,

because thereafter, changing vendors will entail costs that may not be worthwhile, since labor costs (engineers) typically exceed hardware and software expenses.

So far, we have presumed that analog computer architectures will be cheaper than the current standard of the von neumann computer architecture for the purpose of simulating natural phenomena. But how exactly is this cheaper? Currently, we cannot say, other than citing what others have speculated.

Obviously there also exists significant costs to developing and manufacturing unproven technology. But, remember when we spoke of a tradeoff between resource expenditure and accuracy? When we speak of costs, we have to consider what we are proposing in contrast to the current norm, and in this regard, if you desire accuracy, currently it's said that you're limited to simulating the effects of small molecules, on supercomputers or other high performance computing (HPC) environments where the installation and operational costs therein are significant.

Considering for instance, Tianhe-1A: initial costs amounted to $88 million dollars, and another $20 million dollars is spent, per year, on power consumption.[(Tianhe-1)] In contrast, Trafton argues that, "[in] one second, a [typical] cell performs about 10 million energy-consuming chemical reactions, which altogether require about one picowatt (one millionth millionth of a watt) of power."[(Trafton)] This energy efficiency therefrom, as Sarpeshkar proposes, is the product of forgoing high precision operations in favor of low precision analog computation with feedback mechanisms. Which, as Sarpeshkar argues, is fundamentally more efficient in terms of energy, time (computation) and space (memory and program installation) than digital computer architectures.

In this manner, even an approximation of the best case scenario leads to a platform that favors experimentation and iteration, by both small and underfunded terms, to billion dollar pharmaceutical companies. In a manner akin to the aforementioned software ecosystem. That is, if you can build upon abstracted systems written by experts, where these abstractions mask complexity, we may then presume that the overall barrier to entry will drop. That is again, if laypersons are able to built upon abstracted systems in a manner that doesn't require an expert understanding or formal education in such (and presumably in a manner akin to my aforementioned https://imager.io story). But on the other end, if the barrier to entry is lower, this implies reduced costs, and therefore, we may likewise see significant cost savings in industries built upon such, perhaps in a manner akin to using "higher level" programming

languages for applicable problems.

Engineering and Science

The field of synthetic biology is one that attempts to unify engineering and biology. For this intersection, we have a foundation that reduces biology to software. With a proper software ecosystem in place, these bottom-up, cumulative processes may give rise to very sophisticated products, and perhaps one day, akin to the emergent phenomena seen in nature itself. Because, such an ecosystem is akin to the aforementioned "force multiplier of the human mind".

But perhaps too, just because software is typically easier to experiment and iterate on. You don't need a lab with specialized knowhow for the equipment, but in this case, perhaps, access to some cloud based compute infrastructure that affords easy and cheap access to the more specialized analog computing hardware. (If I happen to get into this, I'd perhaps call it SubSystems, given my sub.systems GTLD.)

References

(Does Thinking Really Hard Burn More Calories?) https://www.scientificamerican.com/article/thinking-hard-calories/

(NVIDIA A100 TENSOR CORE GPU) https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet.pdf

(The Thermodynamics of Brains and Computers) https://webhome.phy.duke.edu/~hsg/363/table-images/brain-vs-computer.html

(Sarpeshkar) Sarpeshkar, Rahul. Analog Supercomputers: From Quantum Atom to Living Body, https://youtu.be/ZycidN_GYo0.

(Hehner) Hehner, Eric & Horspool, R.. (1979). A New Representation of the Rational Numbers for Fast Easy Arithmetic. SIAM J. Comput.. 8. 124-134. 10.1137/0208011.

(O'Donnell) O'Donnell, Kevin. (2018). Digital computer simulation of an electronic analog computer. 1-7. 10.1109/LISAT.2018.8378025.

(Achour) Achour, Sara & Rinard, Martin. (2020). Noise-Aware Dynamical System Compilation for Analog Devices with Legno. 149-166. 10.1145/3373376.3378449.

(The AI Hardware Problem) The AI Hardware Problem. YouTube, https://youtu.be/owe9cPEdm7k.

(Grand View Research, Inc) Computational Biology Market Size Worth $13.6 Billion By 2026: Grand View Research, Inc. https://www.prnewswire.com/news-releases/computational-biology-market-size-worth-13-6-billion-by-2026-grand-view-research-inc-300865211.html.

(Flamholz) Flamholz, Avi et al. "The quantified cell." Molecular biology of the cell vol. 25,22 (2014): 3497-500. doi:10.1091/mbc.E14-09-1347

(Trafton) Anne Trafton (Massachusetts Institute of Technology), Cell-inspired electronics. https://phys.org/news/2010-02-cell-inspired-electronics.html.

(Tianhe-1) Tianhe-1, Wikipedia, https://en.wikipedia.org/wiki/Tianhe-1#Tianhe-1A.

(Bernard) Dr. Bernard, A Future Of Medicine. Heme Review, https://youtu.be/iVt5BpoTHYg.

(Francesco) Caravelli, Francesco, and Juan Carbajal. “Memristors for the Curious Outsiders.” Technologies 6.4 (2018): 118. Crossref. Web.

(Siegelmann) Hava T. Siegelmann. 1999. Neural networks and analog computation: beyond the Turing limit. Birkhauser Boston Inc.

(Kish) Kish, Laszlo. (2003). Quantum Computing with Analog Circuits: Hilbert Space Computing. Proceedings of SPIE - The International Society for Optical Engineering. 5055. 10.1117/12.497438.

(MacLennan) MacLennan, Bruce (2007). A review of analog computing. Technical Report CS-07-601, Department of Electrical Engineering & Computer Science. University of Tennessee, Knoxville

(Tucker) John V. Tucker and Jeffery I. Zucker. 2007. Computability of analog networks. Theor. Comput. Sci. 371, 1–2 (February, 2007), 115–146. DOI:https://doi.org/10.1016/j.tcs.2006.10.018

(Sauro) Sauro, H., Kim, K. It’s an analog world. Nature 497, 572–573 (2013). https://doi.org/10.1038/nature12246

(Sarpeshkar) Sarpeshkar, R. “Analog synthetic biology.” Philosophical transactions. Series A, Mathematical, physical, and engineering sciences vol. 372,2012 20130110. 24 Feb. 2014, doi:10.1098/rsta.2013.0110

(Kravitz) Kravitz, Richard L et al. “Evidence-based medicine, heterogeneity of treatment effects, and the trouble with averages.” The Milbank quarterly vol. 82,4 (2004): 661-87. doi:10.1111/j.0887-378X.2004.00327.x

(Siegelmann) Siegelmann, H T. “Computation beyond the turing limit.” Science (New York, N.Y.) vol. 268,5210 (1995): 545-8. doi:10.1126/science.268.5210.545

(Tyson) Tyson, John & Albert, Reka & Goldbeter, Albert & Ruoff, Peter & Sible, Jill. (2008). Biological switches and clocks. Journal of the Royal Society, Interface / the Royal Society. 5 Suppl 1. S1-8. 10.1098/rsif.2008.0179.focus.

(Achour) Achour, Sara & Rinard, Martin. (2020). Noise-Aware Dynamical System Compilation for Analog Devices with Legno. 149-166. 10.1145/3373376.3378449.