Non-Computability and Intractability: Does It Matter to Physics?

Joseph F. Traub
Department of Computer Science
Columbia University

March 4, 1997

Should the impossibility results of theoretical computer science be of concern to physics?

This article lies at the intersection of computational physics and theoretical computer science. Over the last 60 years there has been a stream of negative results announcing undecidability, non-computability, and intractability. Are these impossibility results relevant to physics? I will discuss two of the negative results and provide arguments regarding their relevance.

A number of physicists and astronomers including Robert Geroch and James Hartle[1], Roger Penrose[2], and John Barrow[3] have been concerned about the occurrence of non-computable numbers in physical theories and in the equations of mathematical physics.

Should non-computability be of concern to physicists? I am not convinced and will present arguments for my skepticism.

Typically, the problems of mathematical physics cannot be analytically solved and we resort to numerical computation. Theoretical computer scientists have *conjectured* that the time required to solve many discrete problems grows exponentially with the number of objects. They've *proved* that the time required to solve many continuous problems grows exponentially in the number of variables. When the resources required to solve a computational problem grow exponentially, we say the problem is *intractable*.

Should intractability be of concern to physicists? I'll argue that the question is open. A question with a similar flavor is whether Gödel's theorem should be of concern to physicists. I believe that the answer to this question is also open and defend my answer elsewhere.

My answers to the questions regarding non-computability and intractability depend on which abstract model of the computer is used. Physicists who have thought about this seem to favor the Turing machine. For example, Penrose[2] devotes some 60 pages to a description of this abstract model of computation and its implications. But there is another model of computation which might be more appropriate.

Should physicists consider alternatives to the Turing machine model of computation? I believe that since real and complex numbers are used in mathematical physics, physicists should consider using the real-number model of computation. I present my arguments in the next section. I'll also provide the reader with a primer on the computational complexity of continuous problems.

Should Physicists Consider Alternatives to the Turing Machine Model of Computation?

A central dogma of computer science is that the Turing machine is *the* appropriate abstraction of a digital computer. I will discuss whether it is the appropriate abstraction when a digital computer is used for scientific computation.

First, I'll introduce the four "worlds" that will play a role; see Figure 1. Above the horizontal line are two real worlds; the world of physical phenomena and the computer world, where simulations are performed.

Below the horizontal line are two formal models; a mathematical model of the physical phenomenon and a model of computation which is an abstraction of a physical computer. *We get to choose both the mathematical model and the model of computation. What type of models should we choose?*

Real-World Phenomena	Computer Simulation
Mathematical Model	Model of Computation

Figure 1: Four Worlds

The mathematical model, which is often continuous, is chosen by the physicist. Continous models range from the dynamical systems of classical physics to the operator equations and path integrals of quantum mechanics. That is, mathematical physics uses number fields such as the real and complex numbers. For simplicity I will refer only to the reals in what follows. It is well-understood that the real numbers are an abstraction. That is, it would take an infinite number of bits to represent a single real number; an infinite number of bits are not available in the universe. Real numbers are utilized because they are a powerful and useful construct. Let us accept that today continuous models are central to mathematical physics and that they will continue to occupy that role for at least the foreseeable future. But the computer is a finite state machine. *What should we do when the continuous mathematical model meets the finite-state machine?*

I will compare and contrast two models of computation: the Turing machine and the real-number model. In the interest of full disclosure I want to tell you that I've always used the real-number model in my work but will do my best to present balanced arguments. I will assume the reader is familiar with the Turing machine as an abstraction of a digital computer. Alan Turing was one of the intellectual giants of the twentieth century who defined this machine model to prove a result from logic[4]. In the real-number model we assume that we can store and perform arithmetic operations and comparisons on real numbers exactly and at unit cost. Of course, this is an abstraction and the test is how useful and close the abstraction is to reality.

The real-number model has a long history. Alexandre Ostrowski uses it in his seminal work on the computational complexity of polynomial evaluation in 1954. I used the real-number model for research on optimal iteration theory in 1964. Shmuel Winograd and Völker Strassen used the real-number model in their seminal work on algebraic complexity in the late sixties. Henryk Woźniakowski and I used it in our 1980 monograph on information-based complexity. Lenore Blum, Michael Shub, and Steven Smale provided a formalization of the real-number model for continuous combinatorial complexity and established the existence of NP-complete problems over the reals.

What are the pros and cons for these two models of computation? I'll begin with the pros of the Turing machine model. It is desirable to use a finite-state abstraction of a finite-state machine. Moreover, the Turing machine's simplicity and economy of description are attractive. Another plus is that it is universal. It is universal in two senses. The first is the Church-Turing thesis, which states that what a Turing machine can compute may be considered a universal definition of computability. (Computability on a Turing machine is equivalent to computability in Church's lambda calculus.) Of course, one cannot prove this thesis; it appeals to our intuitive notion of computability. It is universal in a second sense. All "reasonable" machines are polynomially equivalent to Turing machines. (Informally, if the minimal time to compute an output on a Turing machine is $T(n)$ for an input of size n and if the minimal time to compute an output on any other machine is $S(n)$, then $T(n)$ and $S(n)$ are polynomially related.) Therefore, one might as well use the Turing machine as the model of computation.

I'm not convinced by the assertion that all reasonable machines are polynomially equivalent to Turing machines, but I'll defer my critique for the cons of the Turing machine. See Table 1 for a summary of the pros of the Turing machine model.

- Desirable to use finite-state model for finite-state machine
- Universal
    - Church-Turing thesis
    - All "reasonable" machines are polynomially equivalent to Turing machines

Table 1: Pros of the Turing Machine Model.

I'll turn to cons of the Turing machine model. I believe it is not natural to use this discrete model in conjunction with continuous mathematical models. Furthermore, estimated running times on a Turing machine are not predictive of scientific computation on digital computers. One reason for this is that scientific computation is usually done with fixed-precision floating point arithmetic. The cost of arithmetic operations is independent of the size of the operands. Turing machine operations depend on number size.

Finally, there are interesting models which are not polynomially equivalent to a Turing machine. Consider the example of a UMRAM. The acronym reveals the important properties of this model of computation. It is a **r**andom **a**ccess **m**achine where **m**ultiplication is a basic operation and memory access and the operations of multiplication and addition can be performed at **u**nit cost. This seems like a reasonable abstraction of a digital computer since multiplication and addition on fixed-precision floating point numbers cost about the same. But the UMRAM is not polynomially equivalent to a Turing machine! (However a RAM, which does not have multiplication as a fundamental operation is polynomially equivalent to a Turing machine.) Using the example of linear programming, Woźniakowski and I[5] showed that the real number model is also not equivalent to the Turing machine.

The cons of the Turing machine are summarized in Table 2.

- Not natural to use a discrete model of computation in conjunction with the continuous models of physics
- Not predictive of running time of scientific computation on a digital computer
- *Not* all "reasonable" machines are equivalent to Turing machines

Table 2: Cons of the Turing Machine Model.

I now turn to the pros of the real-number model. As I've stated above, the mathematical models of physics are continuous and use real (and complex) numbers. That is, physicists assume a continuum. It seems natural to me to use the real numbers in analyzing the numerical solution of the problems of mathematical physics on a digital computer. For example, investigation of the computational complexity of path integrals has recently been initiated by Greg Wasilkowski and Woźniakowski[6]. They use a real-number model; I believe a Turing machine model would not be natural.

Most scientific computation uses fixed-precision floating point arithmetic. Modulo stability, computational complexity in the real number model is the same as for fixed-precision floating point. Therefore the real-number model is predictive of running times for scientific computation.

A third reason for using the real-number model is that it permits the full power of continuous mathematics. We'll see one example below when I discuss a result on non-computable numbers and its possible implications for physical theories. Using Turing machines the result takes a substantial part of a monograph to prove. With analysis, an analogous result is established in a page.

The argument for using the power of analysis is already made in 1948 by John von Neumann, one of the leading mathematical physicists of the century and a father of the digital computer. In his Hixon Symposium lecture[7], von Neumann argues for a "more specifically analytical theory of automata and of information." He writes:

"There exists today a very elaborate system of formal logic, and specifically, of logic as applied to mathematics. This is a discipline with many good sides, but also serious weaknesses. . . Everybody who has worked in formal logic will confirm that it is one of the technically most refractory parts of mathematics. The reason for this is that it deals with rigid, all-or-none concepts, and has very little contact with the continuous concept of the real or of the complex number, that is, with mathematical analysis. Yet analysis is the technically most successful and best-elaborated part of mathematics... The theory of automata, of the digital, all-or-none type as discussed up to now, is certainly a chapter in formal logic. It would, therefore, seem that it will have to share this unattractive property of formal logic." These observations may be used *mutatis mutandis* as an argument for the real-number model.

An eloquent argument for the real number model is given in the "Manifesto" by Blum, Felipe Cucker, Shub, and Smale[8]. They write: "Our point of view is that the Turing model... is fundamentally inadequate for giving a foundation to the theory of modern scientific computation."

The pros of the real number model are summarized in Table 3.

- "Natural" for continuous mathematical models
- Predictive of computer performance on scientific problems
- Utilizes the power of continuous mathematics

Table 3: Pros of the real-number model.

The con of the real-number model is that the digital representation of real numbers does not exist in the real world. Even a single real number would require infinite resources to represent exactly. Thus the real-number model is not finistic. The Turing machine is also not finistic since it utilizes an unbounded tape. It is therefore *potentially* infinite. Thus, to paraphrase George Orwell, the Turing machine model is less infinite than the real-number model. It would be attractive to have a finite model of computation. (The Turing machine is discrete but unbounded.) There are finite models, such as circuit models and linear bounded automata, but they are special-purpose.

The con of the real-number model is given in Table 4.

- The real-number model is infinite; it is preferable to use a finite-state abstraction of a finite-state machine.

Table 4: Con of the real-number model.

Since my answers concerning non-computability and intractability depend on concepts and results from information-based complexity, I'll provide a very brief introduction and also show how it relates to the rest of computational complexity. See the monographs by Wasilkowski, Woźniakowski and me[9], Arthur Werschulz[10], and Leszek Plaskota[11], and expository papers[12−15] for more material on information-based complexity, which I'll abbreviate as IBC.

Computational complexity measures the minimal computational resources required to solve a mathematically posed problem. For brevity, I'll often use "complexity". The resource I'll be concerned with is time. Consider all possible algorithms for solving a problem; those known and those existing only in principle. The complexity is the minimal cost over all possible algorithms.

Computational complexity may be split into combinatorial complexity and information-based complexity; see Figure 2.

Computational Complexity

$\swarrow$ $\searrow$

Combinatorial Complexity | Information-based Complexity (IBC)

Figure 2: Schema of Computational Complexity

A typical combinatorial problem is the well-known Travelling Salesman Problem (TSP). The input is the location of n cities and the desired output is the minimal route; the city locations are usually represented by a finite number of bits. The complexity of this problem is unknown but almost everyone believes that it is exponential in the number of cities. A problem whose complexity grows exponentially with the "size" of the input is said to be computationally *intractable*. This means that the problem cannot be solved in principle. I'm not considering here possible new forms of computing such as quantum computers. It is *conjectured* that many combinatorial problems are intractable.

I'll contrast this with IBC. Typical problems are high-dimensional integration, path integration, ordinary and partial differential equations, and nonlinear optimization; that is, the problems of scientific computation. Let's consider an initial-value partial differential equation. Typically the initial value is given by a function; it cannot be entered into a digital computer. We discretize the initial value by, say, sampling it at a finite number of points. Thus the information the computer has about the actual mathematical problem is *partial*. The complexity of mathematical models with partial information is studied in IBC. (In particular, continuous models have only partial information.) Since the computer doesn't known the actual mathematical problem we can't hope to solve it exactly—the best we can hope for is an ε-approximation. As is appropriate for scientific computation, IBC uses the real number model of computation.

It has been *proven* that the complexity of most multivariate problems studied in IBC is exponential in the number of variables. They are known to be computationally intractable. The reason we know the complexity of continuous problems but not of discrete problems is that partial information permits us to argue at the "information level".

The intractability results are for the worst-case setting. That is, we require an error at most ε for every input (in some class of inputs). The only chance for breaking intractability is by replacing the worst-case guarantee by a stochastic assurance.

I'll mention two stochastic settings here and illustrate the ideas with a particular example, high-dimensional integration. In Monte Carlo the expected error, with respect to the distribution on the sample points, is less than ε. Then the computational complexity is independent of the dimension; intractability has been broken.

A second setting is the average case deterministic setting. Assume a Wiener measure on the continuous functions. The stochastic guarantee is that the expected error is less than ε. By discovering a relation between this problem and number theory, Woźniakowski in 1991 obtained the complexity of multivariate integration on the average. Experimentation on 360-dimensional integrals arising in mathematical finance by Anargyros Papageorgiou and me indicated that the deterministic methods consistently beat Monte Carlo for the high dimensional integrals of mathematical finance.

So we have very good news. For high dimensional integration intractability can be broken by weakening the worst-case guarantee to a stochastic assurance. But, unfortunately, there are other mathematical problems which remain intractable no matter how we weaken the assurance. Examples are provided by certain Fredholm integral equations and by the approximation problem[10].

Should Non-Computability be of Concern to Physicists?

I remind you that a number is *computable* if there is a mechanical procedure for approximating it to arbitrary precision; see, for example, Turing[4] or Geroch and Hartle[1]. An example of a computable number is π. However, most real numbers are non-computable. A number of physicists have expressed surprise and concern about non-computable numbers in physics. In their seminal paper, Geroch and Hartle[1] ask whether the occurrence of a non-computable but measurable number in a physical theory indicates a difficulty with the theory. Penrose[2] is concerned by the result that the wave equation with computable initial conditions can have non-computable solutions; he calls this a "rather startling result." Faced with the same result Barrow[3] concludes: "The answer to these difficulties, if they can be found, surely lie in an enlarged concept of what we mean by a computation."

It seems to me that there are two issues with respect to non-computable numbers in physics:

- Is it an impediment to comparing experiment with theory?
- Does it indicate a flaw in a physical theory if measurable observables are non-computable?

I'll first consider the question of agreement of theoretical predictions with experiment. Although experimental resuls are known to only limited accuracy, computability is an asymptotic concept. Non-computability does not affect any fixed finite number of digits.

What does matter in drawing conclusions from theoretical models is the computational complexity of computing the i^{th} digit, that is the minimal cost of computing the i^{th} digit. Assume, for example, that computing the i^{th} digit of x must cost 10^{10i} operations. We will never be able to compute more than the first few digits of x even if x is a computable number. Thus, a possible impediment to comparing theory with experiment is computational complexity rather than non-computability.

To illuminate the second question, we discuss several examples. First, consider the paper by Geroch and Hartle. They define measurability and computability very generally and then consider a particular observable in quantum gravity. They present arguments to suggest a certain observable may be non-computable. They also discuss why they are quite far from proving this observable is non-computable.

Next, consider partial differential equations with computable initial conditions but non-computable solutions. The equations are very simple. Examples are the wave equation and the backwards heat equation. The wave equation is assumed to have initial conditions which are not twice differentiable.

These partial differential equations are special cases of ill-posed problems. Recall the definition of ill-posed problems, in the sense of Hadamard. If we seek to compute Lu where L is a linear operator then the problem is said to be ill-posed iff L is unbounded. Marian Pour-El and Jonathan Richards[16] showed that if a problem is ill-posed, then computable inputs might be take into non-computable outputs. They devote a large part of a monograph to prove this result using computability theory.

An analogous result using information-based complexity over the reals was established by Werschulz[17]. Werschulz's proof utilizes the power of analysis and is about one page in length. His approach has several other advantages, as we shall see.

Although Werschulz's result holds on normed linear spaces, for simplicity I'll describe it for function spaces. He assumes that the function u (in the case of differential equations this might be the initial condition) cannot be entered into a digital computer. He discretizes u by evaluating it at a discrete number of points. Werschulz proves that if the problem is ill-posed it is impossible to compute an ε-approximation to the solution at finite cost even for arbitrarily large ε. Thus the problem is unsolvable. Note that this is a much stronger result than non-computability.

But the best is yet to come. In information-based complexity it is natural to consider the average case. The following surprising result was recently established. Every ill-posed problem is well-posed on the average for every Gaussian measure. The measure here is on the inputs u, *e.g.,* on the initial conditions. Werschulz and I[18] surveyed the work leading to this result. We see that the non-solvability of ill-posed problems is a worst-case phenomenon. It melts away in the average case for reasonable measures.

A number of physicists have told me of their unease with the occurrence of non-computable numbers in physical theories, but couldn't give convincing reasons. I'm not convinced that non-computability need be of concern.

Should Intractability be of Concern to Physicists?

As we've seen, the complexity of many discrete problems is *conjectured* to grow exponentially with the number of objects, while the complexity of many continuous problems is *known* to grow exponentially with dimension. These negative conjectures and theorems are for the worst case. Although some continuous problems become tractable if we are willing to live with a stochastic assurance of computing an approximate solution there are others that remain stubbornly intractable.

Many problems of computational physics involve large numbers of objects or variables. Might intractability set fundamental impediments?

Perhaps not. Scientific questions do not come equipped with a mathematical model. Examples of scientific questions include:

- Will there be major climate changes due to human activities?
- Will the universe stop expanding?
- How do physical processes in the brain give rise to subjective experience?

For a scientific question we get to choose the mathematical model. To rigorously demonstrate an impediment due to intractability we should show that *every* mathematical model that captures the essence of a scientific question is intractable. This may be a possible attack in principle but it is far from evident that it could actually be carried out for any non-trivial question. Note, however, that in establishing the computational complexity of a mathematical model we do permit *all* possible algorithms to compete.

Based on our current knowledge I feel that the question stated as the title of this section is open.

The research reported here was supported in part by the National Science Foundation. I appreciate the comments of J.B. Altzman on the manuscript.

References:

1. R. Geroch, J.B. Hartle, *Foundations Phys.*, **16**, 533 (1986)
2. R. Penrose, *The Emporer's New Mind*, Oxford Univ. Press, (1989).
3. J.D. Barrow, *Theories of Everything*, Oxford Univ. Press, (1990).
4. A.M. Turing, *Proc. London Math. Soc.*, **42**, 230 (1937).
5. J.F. Traub, H. Woźniakowski, *Oper. Res. Lett.*, **1**, 591 (1982).
6. G.W. Wasilkowski, H. Woźniakowski, *J. Math. Phys.*, **37**, 2071 (1996).
7. J. von Neumann, *Collected Works, V*, A. Taub, ed. MacMillan, (1963).
8. L. Blum, F. Cucker, M. Shub, M. Smale, *Complexity and Real Computation: A Manifesto*, TR-95-042, Intern. Comp. Sci. Inst., Berkeley, (1995).
9. J.F. Traub, G.W. Wasilkowski, H. Woźniakowski, *Information-Based Complexity*, Academic Press, 1988.
10. A.G. Werschulz, *The Computational Complexity of Differential and Integral Equations: An Information-Based Approach*, Oxford Univ. Press, (1991).
11. L. Plaskota, *Noisy Information and Computational Complexity*, Cambridge Univ. Press, (1996).
12. J.F. Traub, in *Twenty-Fifth Anniversary Symposium, School of Computer Science, Carnegie-Mellon University.* R.F. Rashid, ed., 489, Addison-Wesley, (1991).
13. J.F. Traub, H. Woźniakowski, *Math. Intelligencer*, **13**, 34 (1991).
14. J.F. Traub, H. Woźniakowski, *Sci. Amer.*, **270**, 102, (1994).
15. J.F. Traub, in *Boundaries and Barriers*, J.L. Casti, A. Karlquist, eds. 238, Addison-Wesley (1996).
16. M.B. Pour-El, J.I. Richards, *Computability in Analysis and Physics*, Springer-Verlag, (1988).
17. A.G. Werschulz, *Numer. Func. Anal.*, **9**, 945 (1987).
18. J.F. Traub, A.G. Werschulz, *Math. Intelligencer*, **16**, 42 (1994).