Available online at www.sciencedirect.com
ScienceDirect
Theoretical Computer Science 371 (2007) 4–19
Theoretical Computer Science
www.elsevier.com/locate/tcs

Can Newtonian systems, bounded in space, time, mass and energy compute all functions?

Edwin J. Beggs[a,*], John V. Tucker[b]

[a] *Department of Mathematics, University of Wales Swansea, Singleton Park, Swansea SA2 8PP, United Kingdom*
[b] *Department of Computer Science, University of Wales Swansea, Singleton Park, Swansea SA2 8PP, United Kingdom*

Abstract

In the theoretical analysis of the physical basis of computation there is a great deal of confusion and controversy (e.g., on the existence of hyper-computers). First, we present a methodology for making a theoretical analysis of computation by physical systems. We focus on the construction and analysis of *simple* examples that are models of *simple* sub-theories of physical theories. Then we illustrate the methodology by presenting a simple example for Newtonian Kinematics, and a critique that leads to a substantial extension of the methodology.

The example proves that for *any* set A of natural numbers there exists a 3-dimensional Newtonian kinematic system M_A, with an infinite family of particles P_n whose total mass is bounded, and whose observable behaviour can decide whether or not $n \in A$ for all $n \in \mathbb{N}$ in constant time. In particular, the example implies that *simple Newtonian kinematic systems that are bounded in space, time, mass and energy can compute all possible sets and functions on discrete data*. The system is a form of marble run and is a model of a small fragment of Newtonian Kinematics.

Next, we use the example to extend the methodology. The marble run shows that a formal theory for computation by physical systems needs strong conditions on the notion of experimental procedure and, specifically, on methods for the construction of equipment. We propose to extend the methodology by defining languages to express experimental procedures and the construction of equipment. We conjecture that the functions computed by experimental computation in Newtonian Kinematics are "equivalent" to those computed by algorithms, i.e. the partial computable functions.

Keywords: Newtonian machines; Computability; Hypercomputation; Experimental computation

1. Introduction

The extraordinary development of the theory of computation since the 1930s has been based on its mathematical abstractions from the physical world of machines and technologies. In consequence, the rise of digital computation has granted Computer Science its intellectual independence from Physics and Electronics. The mathematical maturity of our abstract theories of computation allow us to look at the physical foundations of computing in new ways. All sorts

* Corresponding author.
E-mail addresses: e.j.beggs@swansea.ac.uk (E.J. Beggs), j.v.tucker@swansea.ac.uk (J.V. Tucker).

0304-3975/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2006.10.010

of questions arise, from a need to understand interfaces between algorithms and physical technologies (e.g., in new problems of quantum information processing, and in old problems of analogue computers), and from a fundamental curiosity about information processing in physical systems.

What is the physical basis of computation? Is there a theory of the physical process of making a computation?

Consider the idea of computing a function using a physical system. Suppose each computation by a physical system is based on an experimental procedure with three stages:
(i) input data x are used to determine initial conditions of the physical system;
(ii) the system operates for a finite or infinite time; and
(iii) output data y are obtained by measuring the observable behaviour of a system. The function f computed by a series of such experiments is the relation $y = f(x)$. We can expect to compute functions on continuous data, such as the set $\mathbb{R}$ of real numbers, as well as functions on discrete data, such as on the set $\mathbb{N}$ of natural numbers. We call this *experimental computation*.

The idea of experimental computation is general and old. Clearly, it depends upon the choice of
(a) a physical system, which may be a part of nature, a machine, or a combination of both, that performs the computation; and
(b) a physical theory, which we use to reason about the physical system's behaviour and the computation.

The idea can be found in modelling physical and biological systems, e.g., in classical wave mechanics [27,28,43, 44,36,41] or in the emergent behaviour of cellular lattices and neural networks [14,33,47]. It can also be found in technologies for designing machines, e.g., analogue computers [31,25,18,13] or quantum computers [22]. We have attempted to classify and survey the basic literature in our [2]. Over several decades and several fields, the questions arise:
What are the functions computable by experiments with a class K of physical systems? How do they compare with the functions computable by algorithms? Do there exist physical systems in the class K that compute more than algorithms and, hence, qualify as hyper-computers?

Computability theory, founded by Church, Turing and Kleene in 1936, is a deep theory for the functions computable by algorithms on discrete data, and it has been extended to continuous data via approximations. At its heart is the concept of an *algorithmic procedure*.

We argue that, in contrast, the idea of experimental computation is not understood: what are the details of the steps (i)–(iii) above and how is the equipment to be built? We have a methodology for investigating experimental computation in a unified and systematic way. By constructing examples and subjecting them to detailed critiques, we show that at the heart of experimental computation are the concepts of
(a) *experimental procedure*, which is closely related to the notion of algorithmic procedure, and
(b) *experimental equipment*, which is closely related to the notion of a hardware description.
Both are determined by physical theories and are in need of systematic theoretical investigation using logical methods.

Kreisel drew attention to the general question "Is physical behaviour computable?" in some well known papers, e.g., Kreisel [17], which have stimulated research in Computable Analysis. Over the years, there have been physical examples (e.g., [20]), undecidability results for classes of dynamical systems (e.g., [26,7]), philosophical speculations on experimental computation (e.g., [24,8,5,10,48]). But the questions above do not yet have *definitive* answers and the quest for non-computable physical behaviour, or for the reasons it does not exist, will not be abandoned easily. Indeed, in the theoretical analysis of the physical basis of computation there is a great deal of confusion and controversy, especially on the issue of the existence of hyper-computers. (See, e.g., our survey in [2].)

Specifically, in this paper we will examine computation by idealised experiments with idealised physical systems. First, we explain a general methodology, based upon four principles, for making a theoretical analysis of computation by physical systems. The problem we address with this method is to be able to examine in minute detail the whole physical process involved in computation. Our idea is to use *simple examples of systems that are models of simple and precisely specified sub-theories of physical theories and have interesting computational behaviour*. Simplicity is needed to put all the concepts, structures and techniques involved in the process under a theoretical microscope.

Secondly, we apply the method to a case study in Newtonian Kinematics. Thus, we present a simple system that is a model for Newtonian Kinematics and analyse its implications. Now, in our [2], we showed there exist kinematic systems, which operate under the theory of (i) Newtonian Kinematics and (ii) Relativistic Kinematics that can decide the membership of *any* subset A of the set $\mathbb{N} = \{0, 1, 2, \ldots\}$ of natural numbers. The systems were 2-dimensional bagatelles with single particles that needed *unbounded* space, time and energies for their computations.

Here we will improve on these bagatelle machines by constructing new Newtonian machines that are marble runs that each require *bounded* space, time, mass and energy to decide $n \in A$ for all n. These examples are *very* simple yet our critique shows them rich in conceptual detail. These complementary results involve the assumption that space can be infinitely divided into smaller and smaller units, a valid assumption in Euclidean geometry and Newtonian mechanics, but not used on the bagatelles. Using the example, we prove the following theorem:

Theorem 1.1. *Let $A \subseteq \mathbb{N}$. There exists a* 3*-dimensional kinematic system with a family of particles P_n for $n = 1, 2, \ldots$ whose observable behaviour decides A. The system operates under Newtonian mechanics and the entire system is bounded in the following sense:*
Space: The system can be contained within a cube.
Time: The system operates in constant time uniformly for all initial input.
Mass: The total mass of all the particles is bounded.
Energy: The energy needed to operate the system is bounded.

Specifically, the system is a marble run for which the following are equivalent: given any $n \in \mathbb{N}$,
(i) $n \in A$.
(ii) *In an experiment, given initial constant velocity of* 1 *the particle P_n leaves and returns to its origin within a known constant time* 3.

At first sight, the example shows that every set and function on $\mathbb{N}$ is computable by simple bounded Newtonian mechanical systems! Given the Church–Turing thesis, this raises the question, *Is the elementary theory of Newtonian Kinematics undesirably strong and, if so, in what ways?* Thus, our next step is to use the methodology and reflect on the concept of experimental computation. The example highlights the ideas of experimental procedure and equipment.

Now, the example encodes any set A into the static structure of a simple piece of Newtonian kinematic equipment such that a simple experimental procedure, based on the observation of the dynamic behaviour of a particle, can recover the set A from the given equipment. This shows that the idea of equipment is problematic when computing with Newtonian mechanics.

Our critique of the example shows that the idea of experimental computation involves much more than an informal experimental procedure applied to a black box containing an arbitrary given physical system. Crucially, it involves
(i) the application of a sequence of *experimental actions* to the marble run, and
(ii) the *process* of *assembly* or *construction* of the equipment.

The distinction between the existence of a black box that is a hyper-computer from its construction by people is made in Beggs and Tucker [2], in Ziegler [49], and more forcibly here.

We propose a new approach of using languages, derived from physical theories, for "programming" both experimental procedures and construction procedures. Our example shows that a refinement of mechanics is needed to define classes of *constructible equipment*, which would restrict the subsets A of $\mathbb{N}$ in the marble run. The design of such languages is complicated. We conjecture that using such a refinement of Newtonian Kinematics the functions computed by experimental computation are "equivalent" to those computed by algorithms, i.e. the partial computable functions.

Thus, our framework considers experimental computation using the combination of "programming" and "hardware description" languages that are derived from physical theories. This is different from computability theory which does not study the construction of its abstract machines and computational models in an explicit and systematic way.

We think examples like the marble run are of interest in their own right. The problems it poses can be analysed by other means; for example, the marble run can be used to explore the interface between a constructive or intuitionistic theory with a physical theory.

The structure of the paper is this. In Section 2 we present our general principles for an investigation of experimental computation. In Section 3 we sketch a small sub-theory of Newtonian Kinematics within which we will construct the example. In Section 4 we specify the construction of the marble run. In Section 5 we propose the languages for experimental procedures and equipment. Finally, in Section 6, we discuss some open problems.

This paper is a sequel to Beggs and Tucker [2], which contains complementary results (that do not involve the infinite division of space but do require unbounded energies) and a literature survey. It is a companion to our [3], where the general principles of Section 2 were first formulated. Technically, only elementary Newtonian mechanics is needed to follow our arguments. However, some knowledge of the different approaches to the formulation of the problem and its "solution" is necessary. Thus, as background, it is helpful if the reader is familiar with [2] and with

the theory of the functions computable by algorithms on discrete data, and its extension to continuous data (see: [28, 35,38,39,42]).

We are indebted to an anonymous referee for several useful comments, which have improved our paper. In particular, he or she has enabled us to include observations on the need for meta-theory in Section 2.2.

2. Method for analysing experimental computation

We are interested in making a *theoretical* analysis of experimental computation. The concept of experimental computation, explained in the introduction, emphasizes that there are *many* things we need to consider when thinking about the physical basis of computation, for example:
(i) the set of data and how it is represented;
(ii) function to be computed on the data;
(iii) the operational procedures necessary to measure, which define the physical representation of the data;
(iv) the operational procedures necessary to initialize, control and observe the course of the experiment in time;
(v) calculations that might be involved in performing the experiment;
(vi) the system and how it is to be designed, created and used as equipment.

In this section we propose a set of basic principles for investigating experimental computation by physical systems.

2.1. Role of physical theories

Physical theories play a complicated role in experimental computation. In seeking answers to the questions in the introduction, we should use a physical theory to

(1) define precisely the class K of physical systems under investigation.
This means that the concepts and laws of the theory will be used to
(2) design and construct the systems in K;
(3) explain and prove properties of their operation and behaviour; and, hence,
(4) validate experimental computations by the systems of K.
For example, a physical theory is needed to design, operate and validate a Newtonian system capable of simulating a Turing machine.

If a hyper-computer has been found in K then we should use the theory to
(5) evaluate the physical credibility of the system or, possibly,
(6) reveal weaknesses in the theory.

Finally, in each case, we should use the theory to
(7) make clear, precise and detailed statements about the whole process of experimental computation, especially about the experimental procedure and equipment.

It is clear that these simple desiderata are not all present in the literature on hyper-computation, though given the controversies, (5) is popular. In the case of hyper-computers, examples can involve complicated theories, ignore aspects of the experimental process, and seek to be more persuasive than informative. Comprehensiveness, clarity, precision and details are in short supply.

In investigations of experimental computation and the questions above, it is the physical theory itself – and, indeed, the process of changing the theory – that is the central object of theoretical study. Our method for an investigation of some type of experimental computation is based on these four principles and stages.

Principle 1 (*Defining A Physical Sub-Theory*). *Define precisely a sub-theory T of a physical theory and examine experimental computation by the systems that are valid models of the sub-theory T.*

All physical theories are large and changing, though the classical theories usually have an elegant mathematical form and can be organised by incrementally adding concepts, laws and techniques. In some cases they have been axiomatised informally and even formally. Clearly, it is not easy to define exactly such a theory. Therefore, one might begin with some theory that is quite large, for example the Newtonian Kinematics of point particles, which is a sub-theory of Newtonian Mechanics. By working on examples, conceptual precision and mathematical rigour will surely

reduce the sub-theory into something *much* smaller. For example, suppose one finds an interesting system C that is a computer or hyper-computer and a model of the physical theory. Then the problem arises:

Minimal sub-theories. *In computing with a physical system C, specify the smallest sub-theory or fragment T of the physical theory that is needed to construct, operate, reason about, and validate C.*

The chosen sub-theory T, whether minimal or not, is the source of the definition of experimental computation. What is contained in T determines the set of possible experimental procedures and what forms the systems have in terms of their components and architectures. Later we discuss how to derive, from a sub-theory T, languages to specify formally both experimental procedures and equipment.

For the purposes of a theoretical programme the idea is to look at simple theories that are small and rather precisely defined. *It does not matter whether we think of them as true, or roughly applicable, or know them to be false.* For example, in all work on Newtonian systems, here and elsewhere, it is *known* that assumptions about space and time are false of our universe. In our view, that in no way diminishes the theoretical interest and value of seeking Newtonian hyper-computers. As we hope to show here, success and failure in the search can provide insight into both the nature of computability in a physical context and the scope of Newtonian principles. If Newtonian examples can be confusing and disputed then it is not surprising that Relativistic and Quantum examples are not definitive.

Principle 2 (*Classifying Computers in a Physical Theory*). *Find systems that are models of T that can, through experimental computation, implement specific algorithms, calculators, computers, universal computers and hyper-computers.*

Sub-theories do not contain notions of computation, though the term *information* is commonly used in some theories since Boltzmann. The computation of functions and sets must be abstracted from, or represented by, the operation of physical systems along the lines mentioned above. For the purposes of comparison with algorithmic procedures, and especially for the search for hyper-computation, there is a natural course of action, namely study the mathematical problem:

Embedding computations in a physical theory. *Seek ways of embedding functions, sets, logical formulae, algorithms, programs, models of computers and hyper-computers, etc. into some physical systems that satisfy a sub-theory T.*

Clearly, as with any embedding problem, we are interested in embedding *complex algorithmic computations* into *simple physical systems* that are models of *simple physical sub-theories*. For example, to determine if there exist Newtonian systems that compute everything, one can embed *arbitrary sets* into valid 2-dimensional single particle systems obeying a few simple Newtonian or Relativistic laws (see [2]). In this paper, we will embed an arbitrary set into a very simple 3-dimensional kinematic system.

At this early stage in our understanding, the idea is to *play* with computational ideas and their embedding into physical theories, in order to explore the interface between algorithms and experiments and to see what our ingenuity or trickery can get away with and what it can uncover. In our experience, and that of others, hyper-computations are not "genuine" but involve the embedding of non-computable sets, functions, etc., into systems, especially in the initial states of systems or parameters. Actually, such embeddings are hidden by the *complexity* of the examples and have to be unearthed. It seems that hyper-computation is courtesy of *deus ex machina*. The literature is plagued by uncertainty and controversy. For this early stage of investigation, *we should not care about the validity of the physical theory but we should care about being able to make theoretically complete models and precise mathematical analyses that reveal all the concepts and technicalities.*

Principle 3 (*Mapping the Border Between Computer and Hyper-Computer in Physical Theory*). *Analyse what properties of the sub-theory T are the source of computable and non-computable behaviour and seek necessary and sufficient conditions for the systems to implement precisely the algorithmically computable functions.*

It is an intriguing and complicated problem to isolate what properties of a physical theory permit computers and hyper-computers. In the case of systems built from embeddings it may be clear where some non-computable resource is to be found. Thus, if we constrain that resource we may be able to recover computability. In our Newtonian work it is obvious where non-computability enters and we can isolate some necessary and sufficient conditions on the sub-theories that can control the embedding of sets of natural numbers. However, they amount to *algorithmic* conditions on

mechanical equipment that are new and external to classical mechanics. This importation is a difficult and fundamental point for the conditions *ought* to belong to the physical theory. What, if any, algorithmic conditions belong to a physical theory? Answers must avoid circularity. For example, for certain arguments it may not be acceptable to assume that standard models of calculators and computers are simply available for use in experimental computation. Strictly speaking they are external or foreign to the physical theory T.

Calculations: *Which of the standard models of calculators can be implemented in the T?*

It is not clear that a Turing machine can be faithfully embedded in many physical theories: see Davies [9]. Indeed, it is an interesting problem to investigate which models of computers can be represented or implemented in a physical theory T.

Principle 4 (*Reviewing and Refining the Physical Theory*). *Determine the physical relevance of the systems of interest by reviewing the valid scope or truth of the sub-theory. Criticism of the system might require strengthening the sub-theory T in different ways.*

We have emphasised the autonomy and independence of the physical theory since we take it and its sub-theories to be the focus of the mathematical work. In addition to providing intellectual pleasure, a physical theory is supposed to be true of some aspect of the physical world. Of course, a physical theory cannot be proved, but it can be validated by experimental facts. For any theory T there are restrictions that define that part of the theory that has been validated experimentally. These restrictions can be seen as a refinement theory $E(T)$, called the *experimental domain* of T where models are supposedly more faithful to experimental results. $E(T)$ will contain numerical bounds on physical measurements, for instance. For any valid model C of a theory T, we may ask:

Experimental validity: *Is C a valid model of the experimental domain $E(T)$ of T?*

The process of reviewing examples can lead to the refinement of the sub-theory in different ways, perhaps adding more laws in the search for more realism — e.g., what happens if we add friction and/or elasticity to the Newtonian system. And it can lead to the rejection of the sub-theory and its systems as a basis for making a hyper-computer — e.g., arbitrary velocities, and hence energies, may be needed, which are not possible. Often, as one moves out of $E(T)$ into T one contradicts other theories — theories created to cover a larger experimental domain. In Newtonian kinematics, using arbitrary velocities contradicts Relativity Theory, and making arbitrary small components contradicts Atomic theory. In our work, we have worked with the relevant full, unrestricted, kinematic theory T, ignoring the experimental domain, and have concentrated on mathematical rigour, consistency and completeness.

2.2. Formulating, axiomatising and changing theories

The methodology above will be illustrated in the next sections. However, we wish to comment on two aspects of the method that will not feature largely in our example in the next section but will require further development as the method is used. These aspects are:
(i) the *formulation* of the physical theory T (e.g., through axioms, postulates, laws, etc.), its models, and the nature of its use in experimental computation (e.g., in calculations, deductions etc.);
(ii) the use of *portfolios* of theories, and the relationships between theories, that arise in the analysis of examples.

The development of any theory is full of puzzles when viewed under a philosophical microscope. For instance, Newtonian mechanics is rich in deep questions for the philosophy of physics and epistemology in general.
Formulation. Consider the formulation of the physical theory T. Physical theories are not easy to define *exactly*. Newtonian Kinematics is huge and contains alternate mutually inconsistent assumptions, e.g., different laws of friction with different experimental domains.

In our framework we have proposed the idea of constructing a minimal sub-theory T to document the essential features of experimental computation with a particular system C. Further work is needed to make the idea precise but perhaps an example will help.

Consider a very simple system C of motion in one dimension, where we fire standard balls (of identical size and composition) vertically up or down at a given velocity, and observe their position at a later time. To say that this system is a model of Newtonian Kinematics is to say that we assume that the system satisfies some of the laws of Newtonian Kinematics. More precisely, let us assume that it satisfies the physical theory T containing only Newton's Laws of Motion and the law of a uniform gravitational field. These laws can be summarised as:

1. Force equals inertial mass times acceleration.
2. Force equals gravitational mass times a constant, g.

In the framework, we say loosely C is a model of T.

In his controversial experiment (the controversy being whether it ever happened), Galileo dropped balls of differing weights from the Leaning Tower of Pisa to test whether heavier balls fall faster than lighter ones. He later formulated the idea that all balls (in the absence of air resistance) should fall at the same rate. A modern way of saying this is the *equivalence of gravitational and inertial mass*. That is, given our ability to absorb a universal constant into g, that inertial mass equals gravitational mass. This has now been tested to high accuracy, but in reality there could be a very small difference in the acceleration of a gold ball and a lead ball in a gravitational field. However in our model the balls are all standard, so we can propose a reduced theory T_M which adequately predicts the results, and which would reasonably seem to be minimal:

3. Acceleration is a constant, g.

The experimental domains would further refine these theories with numerical bounds.

Axioms. The mathematical development of mechanics spans many centuries and contains many landmarks in the pursuit of, generality, precision and logical structure. One example is Newton's use of calculus to derive Kepler's Laws from the Law of Gravitation and his Laws of Motion. Another example is the logical and algebraic structure of Lagrange's *Mechanique analytique* of 1788. Physicists are not primarily interested in deduction, of course. So, since our problem of experimental computation is located at the interface between physics and logic, and our method requires high precision, it will involve investigations of physical theories using logical and philosophical methods. For example, it may well require formal axiomatisations of some small theories to bring them within the scope of proof and model theory for (say) first order theories. Such logical investigations of physical theories are not new and have been studied in some depth by M. A. Bunge, see Bunge [4], for instance. Thus, the formalisation of small theories designed to explain particular examples seems tractable.

Portfolios. Now, consider the need to look at variations of the physical theory T. In analysing an example, we are likely to encounter not just one theory but a family or *portfolio* of theories. For example, if we specify a system with a special computational property neglecting friction then later we may need to check that the basic property is preserved if some form of friction is added. (This happens in the marble run that follows.) Also, we have already talked of the minimal theory needed to define a model C and of the experimental domain $E(T)$ of a theory. In dealing with portfolios of theories, we are likely to enter the field of the theory of physical theories or *meta-theory*.

We will need to understand the relationship between sub-theories. As with axiomatics and deduction, meta-theory is not new, for physics is greatly interested in understanding the relationships between its theories. Among a number of accounts these may be relevant:

(a) There is a "global" and "evolutionary" approach to meta-theory in which different theories evolve, compete and may be unified, possibly into grand theories, such as in the case of the unifications of Magnetism and Electricity by Faraday and Maxwell, Electrodynamics and Gravity by Einstein, and Electrodynamics and Quantum Theory by Feynman. For example, C.-F. von Weizsäcker expresses the view that different theories evolve toward a final Grand Unified Theory describing nature in principle [45,46]. The "space" of theories has some ordered structure and may be modelled by some kind of category or lattice, which should help to define and justify the existence of minimal or irreducible theories.

(b) There is a "local" approach to meta-theory in which different theories have different domains of application. There are overlaps and the assumption that each aspect of nature is covered by some theory. But theories have different applications. Here there is an analogy with open sets or charts of a manifold. See Schroeter [32] and Ludwig [19].

The picture is very complicated. In working in the framework of kinematics we may easily depend on other theories (materials, optics, sound, etc.) when we justify some observational technique. These auxiliary theories are called pre-theories in Schroeter [32].

Other studies, emphasising mathematical theories are Stegmueller [37] and Balzer et al. [1].

3. Theory: Experimental computation with a small sub-theory of Newtonian kinematics

In using the framework, we begin the study of the marble run by reflecting on the underlying theory. Newtonian Kinematics is about systems of particles in n-dimensional space ($n = 1, 2, 3$) satisfying Newton's Laws of Motion and

laws such as Gravitation and Conservation of Potential and Kinetic Energy. Typically, sub-theories describe systems that allow the
(a) projection of particles in space with arbitrary velocity, and
(b) observation of the position of particles at certain times.

A sub-theory of Newtonian kinematics may further assume that the space may be structured in different ways, influenced by gravitational fields, populated by special bodies and obstacles, and shaped by geometries. It may also suppose that materials have different physical properties such as friction and elasticity. It may also require certain numerical calculations to be made.

Our example will need a simple fragment of Newtonian Kinematics in which the basic needs for the marble run are inertia (so that the ball keeps moving) and reflective barriers. In summary, we will need to place, project and observe the position of a particle in space, and measure time on a clock. We will *not* need absolute precision in measurements, either in space or time; in fact, we can allow generous margins of error in measurements. Only in the initial placement of a ball on a track must some care be taken. We will *not* need to calculate with algebraic formulae.

We can try to postulate a minimal set of laws needed for the operation of this particular machine.

Definition 3.1. Let T_M be a set of components and laws consisting of the following:
1. The ability to project a ball from a given position with an (approximately) given velocity.
2. The ability to observe a ball.
3. The ability to measure time.
4. Newton's three Laws of Motion.
5. Elastic barriers to reflect the ball.
6. Gravity, to constrain the particles in grooves, and to make them fall into a collecting box.

Of course the details of the machine can be varied and other minimal theories described: see 4.2.

4. Example: The marble run

4.1. Proof of main Theorem 1.1

Experiments with the marble run. The marble run is a surface, lying within a square, with tracks or groves along which particles can run. At both ends of the tracks are trays to catch particles. The experimental procedure is this:

To find if $n \in A$, take the particle P_n and fire it along the track indexed n with velocity 1. If the particle returns within time 3, then $n \in A$, if not then $n \notin A$.

The reason for having specific particles P_n for different n is that the tracks on which the particles run get narrower as n increases. The particle P_n is defined by its mass $M(n)$ and radius $R(n)$ which is calculated from n. Assuming a common material of fixed density it is enough to calculate a radius $R(n)$ when choosing a particle.

Structure of the marble run. Consider a metal plate forming a quarter annulus, which we take to run from angle $\theta = 0$ to $\theta = \pi/2$ anticlockwise from the centre. The inner radius of the quarter annulus is 1, and its outer radius is 2. Inside radius 1 and outside radius 2 there are deep trays into which the particles may fall; these are called the *inner* and *outer trays*, respectively.

On the annular metal plate are cut infinitely many radial tracks labelled by the set $\mathbb{N}$. Beginning at $\theta_0 = \pi/4$, the tracks are placed by bisecting angles in the anticlockwise direction, at angles

$$\theta_1 = \theta_0 + \pi/8, \qquad \theta_2 = \theta_1 + \pi/16, \ldots, \qquad \theta_n = \theta_{n-1} + \pi/2^{n+2}.$$

To be precise, the centre line for each track n is a radius from the centre of the annulus at the angle

$$\theta_n = \frac{\pi}{2}\left(1 - \frac{1}{2^{n+1}}\right).$$

The distance between the centre lines of track n and $n+1$ measured along the inside circle is $\pi/2^{n+3}$. The width of track n is $\pi/2^{n+6}$, and that of track $n+1$ is $\pi/2^{n+7}$, i.e., half the width of track n. So the tracks do not overlap.

The polar coordinates of the starting point of track n is $X(n) = (1, \theta_n)$.

Each particle P_n has diameter equal to the width $\pi/2^{n+6}$ of track n so its radius is $R(n) = \pi/2^{n+7}$.

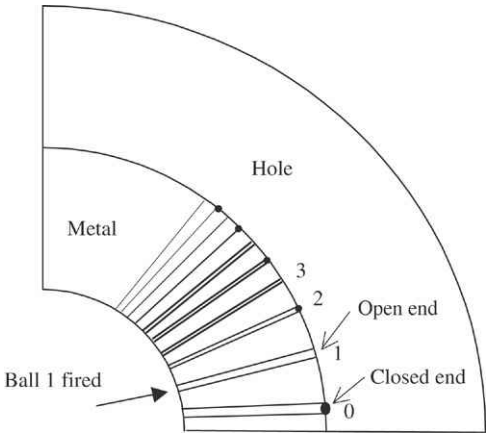

Fig. 1. The marble run.

At the end of track n is either a spring to return the particle, for $n \in A$, whence it falls into the inner tray, or an open end, for $n \notin A$, which lets the particle fall down into the outer tray.

The marble run corresponding to the subset of even natural numbers is illustrated in Fig. 1, with the first 4 tracks labelled and the springs marked by closed ends. As the track numbers for the first 4 natural numbers increase the tracks become progressively narrower and closer together. The figure is expanded with tracks beyond $\pi/4$ for clarity.

Operation of the marble run. Particle n rolls at speed 1 to the end of the track in time 1. Then it either falls in the tray, never to return, or hits the spring. For a reasonably good spring we assume that it is reflected at a velocity, say, $>3/4$. The return time for ball n if the end of track n is closed is $<7/3$. By checking the inner tray for a returned particle at time 3 we build in the possibility of some margin of error in the timekeeping, we do not require arbitrarily high precision.

Bounds. The formulae above gives all the infinitely many tracks fitting into the quarter annulus, so the machine fits into a bounded space, namely a box of size $3 \times 3 \times 1$.

The masses (taking each ball to be of the same density) of each ball are bounded, and by summing the masses, the total mass required is also bounded. For if the density of each particle is ρ, and each particle P_n has diameter $2R(n)$ equal to the width $\pi/2^{n+6}$ of its track n, the mass $M(n)$ of the particle P_n is

$$\frac{4}{3}\pi R(n)^3\rho = \frac{4}{3}\frac{\pi^4\rho}{2^{3n+21}}.$$

This gives the total mass of all the balls for $n \geq 0$ to be

$$\sum_{n=0}^{\infty}\frac{4}{3}\frac{\pi^4\rho}{2^{3n+21}} = \frac{4}{3}\frac{\pi^4\rho}{2^{21}}\sum_{n=0}^{\infty}\frac{1}{8^n} = \frac{\rho\pi^4}{3\times 2^{16}\times 7}$$

since the geometric series of common ratio $\frac{1}{8}$ has sum $\frac{1}{1-\frac{1}{8}} = \frac{8}{7}$.

The velocities used are bounded above by 1, and together with the mass bound this shows that the kinetic energy required is bounded by the kinetic energy of the largest particle P_0, i.e.

$$\frac{1}{2}M(0)1^2 = \frac{\rho\pi^4}{3\times 2^{20}}.$$

This completes the proof of the theorem.

Finally, by taking A to be the coded graph G_f of any function f, we have the corollary:

Corollary 4.1. *Any function $f:\mathbb{N}\to\mathbb{N}$ can be computed by a Newtonian marble run.*

4.2. Variations of the marble run

Variations of the marble run lead to variations in the physical theory. Both may be simplified or made more complicated. For example, if we work with the simpler assumption of point particles then the complications of selecting different size balls can be dispensed with. On the other hand, for convenience we have assumed that there is no friction: if it was included we would need to make adjustments to the speeds of the particles. Another complication is to have some energy loss (a degree of inelasticity) on hitting the barrier. We could replace the reflecting barrier with an incline, so the machine could be modified so that rule (3) in Definition 3.1 is no longer necessary — we would have reduced the list of rules. Likewise, in the absence of gravity, as in rule (4) in Definition 3.1, we could replace the groove with a tube.

The desire to include these and further physical properties leads to a portfolio of related physical theories.

5. Critique: Languages for specifying experimental procedures and equipment

We will consider experimental computation using the marble run and extend the framework discussed in Section 2. First, let us check on the sub-theory presented in Section 3.

5.1. The sub-theory and the marble run procedure

The procedure given at the beginning of Section 4 seems very simple. Now, given any marble run M_A, the experimental procedure needed to decide $n \in A$ can be carried out using the following *primitive experimental actions*:
(i) choose and place a particle in space;
(ii) project a particle in space;
(iii) observe the position of a particle in space;
(iv) measure time on a clock.

There are some parameters defining the mass, size and velocity of any particle, but, in the case of the marble run, the actions required are simple and are based on T_M in Definition 3.1. Notice that we can also tolerate inaccuracies in measurements and that we did not calculate with even simple algebraic formulae.

To define this minimal theory T_M, on which specification and reasoning about the system M_A is based, fully and formally is quite a task. We can make T_M a little more precise by trying to formalise the procedure as follows.

Given any marble run M_A, we can express the experimental procedure in the following "experimental pseudo-code":

exp-pseudo-code Marble run;
place particle mass $M(n)$ radius $R(n)$ at point $X(n)$;
start clock t;
project particle at $X(n)$ with velocity 1;
wait 3 units and do nothing;
if particle in inner tray **then** return "$n \in A$"**else** return "$n \notin A$"
end Marble run.

The simple primitive actions of T_A become instructions and their combination by control constructs result in programs expressing experimental procedures. Indeed, there is a whole language L_M for the pseudo-code to be derived by T_M. Said differently, the semantics of the pseudo-code language L_M and its procedures are given by T_M.

What is the relationship between the equipment – the hardware – of M_A and the minimal theory T_M?

First, we note that the marble run M_A satisfies the laws of T_M, i.e., M_A is a model of T_M. The structure of the equipment M_A is based on an Euclidean construction determined by the set A of numbers. Since Newtonian mechanics is based on Euclidean geometry there is *nothing* in Newtonian mechanics that prevents or even cautions us from defining such systems for *any* set A. A physical theory such as T_M includes Euclidean geometry and, normally, places no conditions on the set A for the marble run M_A to be a valid model of T_M. However, when we use it to compute, we get the extreme result that Newtonian systems *exist* to compute any set or function on $\mathbb{N}$.

Should the marble run M_A qualify as a legal piece of Newtonian equipment for experimental computation and if not, why not?

We answer "No". But the reasons present us with some interesting open problems that we will now discuss.

5.2. General reflections on experimental computation and the role of equipment

Our idea of computing by experiment with the marble run is, roughly, this:
Provisional definition. Experimental computation consists of an *experimental procedure* which is a specification for a finite or infinite process made of primitive experimental actions that are applied to a physical system called the *experimental equipment*. The actions include initialisation and observation of the equipment, which is otherwise considered to be a black box. The procedure schedules the actions in time, step by step, using one or more clocks. Briefly,

$$\textit{Experimental computation} = \textit{Experimental procedure} + \textit{Equipment}.$$

To understand this notion of experimental computation we must analyse the ideas of experimental procedure and equipment, and how they may be combined. We expect the notion of experimental procedure to be well behaved but our theorems here and elsewhere demonstrate that there will be problems with the notion of equipment. This leads to a revised definition shortly.

5.2.1. Experimental procedures

Inspired by experimental procedures expressed in pseudo-code (Section 5.1), we propose the idea of associating with a physical theory T certain languages for expressing experimental procedures. Each language $\mathrm{EP}(T)$ is similar to an imperative programming language in which the basic instructions are:
(i) primitive experimental actions;
(ii) simple algebraic calculations; and
(iii) simple control constructs.

A physical theory T is needed to provide a semantics for the instructions and procedure, e.g., to validate the physical instructions of the language. Briefly,

$$\textit{Experimental procedure} = \textit{Experimental actions} + \textit{Calculations} + \textit{Control and Scheduling in Time}.$$

Specifically, the actions or basic instructions of $\mathrm{EP}(T)$ are derived from a fragment T of a physical theory. The calculations are terms over a simple data type of numbers and the control constructs are standard conditionals and iterations. We have begun to explore this idea and will report on our work in a paper shortly.

5.2.2. Equipment

Suppose we start by defining experimental equipment to be simply a physical system to which the instructions of the experimental procedure can be applied in sequence. Thus, the idea of *Newtonian equipment* is essentially any system for which it is meaningful to apply the primitive instructions and the laws of Newtonian mechanics. More specifically, it is a black box and is governed by physical laws of a sub-theory T, as assumed in our provisional definition.

However, now suppose the specification of an experimental computation is required to explain how the equipment is *constructed*, as a step-by-step process taking place *in time*. We have an interesting problem. The marble run is finite in its materials and operation but not in its architecture and construction. If the marble run is not acceptable then plenty of systems that are valid models of Newtonian mechanics are to be declared invalid, at least for the purposes of experimental computation, if not in general. But, what *exactly* are the new conditions that specify the idea of legal or valid Newtonian equipment? Do these conditions already belong to Newtonian Mechanics or are they new?

The marble run suggests the need for a notion of *construction procedure* for equipment based on *primitive construction steps*. For the marble run, the primitive construction steps involve simply:
(i) cutting tracks and holes;
(ii) placing springs;
(iii) calibration.
These actions are applied according to a blueprint that is made using the
(iv) primitive ruler and compass operations of Euclidean geometry.
The construction also involves:
(v) a uniform gravitational field,
(vi) materials with negligible friction.

For Newtonian equipment *in general* there is a much wider range of primitive construction actions to consider. For example, the steps involving the geometry of the space need not be limited to those generated by ruler and compass operations. There are many choices for the (i) devices, (ii) obstacles and (iii) force fields placed in the space. We can also choose materials with various densities and different forms of friction and elasticity.

A choice of primitive actions or steps based on a theory T leads to a set of instructions for a *construction language* $\mathrm{CP}(T)$ for expressing construction procedures. We define Newtonian equipment to be *constructible* in the construction language $\mathrm{CP}(T)$ if it is definable by a construction procedure of $\mathrm{CP}(T)$. Briefly,

$$\textit{Construction procedure} = \textit{Geometry} + \textit{Devices} + \textit{Force Fields} + \textit{Materials}.$$

Clearly, construction languages allow us to place conditions on the form of equipment and explain why the marble run might not be valid in a theory of experimental computation. Case studies of languages are the subject of our on-going work.

5.2.3. Integration of experimental and construction procedures

Finally, we need to comment on how the instructions of a construction language $\mathrm{CP}(T)$ are combined with the instructions of an experimental procedure language $\mathrm{EP}(T)$ to form a complete language $\mathrm{ECP}(T)$, wherein

$$\mathrm{EP}(T) \subset ECP(T) \quad \text{and} \quad \mathrm{CP}(T) \subset ECP(T),$$

for experimental computation and to argue why such a language cannot express the marble run.

In a procedure from $\mathrm{ECP}(T)$ which gives a complete specification of an experimental computation there are two cases:

(i) *Independent or total equipment specification*. All the steps of the construction procedure can be completed *before* starting the steps of an experimental procedure. The equipment is independent of the experimental procedure.

(ii) *Dependent or partial equipment specification*. The steps of the construction procedure are interleaved with steps of an experimental procedure. The equipment and the experimental procedure are dependent on one another.

In case (i), the equipment involves a finite number of construction steps. It is possible to complete the building of the equipment in finite time and deliver it ready to perform an experimental procedure on all data. The equipment is finite in the sense that its parts – as determined by the instructions of the construction sub-procedure executed in its manufacture – are finitely many.

For any equipment that involves a potentially infinite number of construction steps the equipment cannot be completed in finite time and delivered to the experimental procedure, and case (i) cannot apply. However, the second case (ii) of partial equipment is possible, for given some input data, the construction steps can be followed to build a *sufficiently large* piece of equipment for the experimental procedure to be performed. In the marble run, given input n we need to have enough of the equipment constructed to include track n.

By including the construction of the marble run in the definition of experimental computation, and given the ideas on languages, in the case of the marble run we must interleave the instructions and procedures in the partial specification and have a "definition" of the form:

Definition 5.1. A is decidable by experiment with the marble run M_A if, and only if, A is decidable by an experimental procedure from $\mathrm{ECP}(T_M)$ applied to M_A and M_A is partially constructible.

(Of course, a problem in the construction seems to be that the sequence of primitive construction steps for the marble run M_A involves knowledge of the set A — indeed precisely the knowledge the system M_A is being designed to reveal, making the purpose of the experiment redundant. But since we are interested in the *nature* of experimental computation, this point about redundancy is not interesting.)

5.2.4. Classification

Consider mapping the border between computer and hyper-computer, according to Principle 3 in 2.1. The pseudo-code is a sketch proof that A is decidable by an experimental procedure from the sub-language $\mathrm{EP}(T_M)$ of $\mathrm{ECP}(T_M)$. So it is the construction of M_A that needs analysis.

What conditions are needed on A to enable us to prove that M_A is partially $\mathrm{CP}(T_M)$ constructible (or semiconstructible), or not?

To build useful parts of the marble run in stages, the construction procedure must be able to decide A or generate finite subsets of A. Now, for *any* reasonable choice of an experimental specification language ECP(T) for any T, the procedures will form a countable set. Thus, not every set A of natural numbers can be decided or generated and, hence, not every marble run M_A is constructible in ECP(T). Thus, under Definition 5.1, marble runs do *not* compute all functions on $\mathbb{N}$!

One idea is to suppose that $A \in \mathbb{N}$ should be algorithmically decidable or semidecidable for use in a construction procedure from ECP(T_M). There is an argument that the marble run M_A is made using an idealised CAD tool that cuts the metal, and the tool needs an algorithm for A in order to do this. If we do introduce such classical computability concepts into ECP(T_M) then we seem to be on the way to a proof of:

Conjecture 5.2. *The following are equivalent.*
(i) *A is decidable (or semidecidable) by experiment with the marble run M_A*
(ii) *A is decidable (or semidecidable) by algorithms.*

However, to introduce classical computability into the physical theory via ECP(T_M) runs a risk of circularity for we are seeking to understand the physical foundations of computability theory. To show (ii) implies (i) one wants a *physical* process, based on T_M or an extension, to generate A. For instance, we need a theorem that shows there is an experimental procedure and equipment, definable in the language ECP(T_M) based on T_M, that defines computable and semicomputable sets. So the conjecture remains for it is a subtle problem to formalise the languages CP(T_M), EP(T_M) and ECP(T_M) and show or refute this.

5.2.5. Revised definition of experimental computation

Our idea of experimental computation is now revised to this:

Definition. *Experimental computation* consists of:
(i) An *experimental procedure* which is a specification for a finite or infinite number of primitive experimental actions that are applied to a physical system called the *experimental equipment*. The actions include initialisation and observation of the equipment.
(ii) A *construction procedure* which is a specification for a finite or infinite number of primitive construction actions that builds the *experimental equipment*.

The procedures can be combined and may depend on one another, and schedule the experimental and constructional instructions in time, step by step, using one or more clocks.

Briefly,

$$\textit{Experimental computation} = \textit{Experimental procedure} + \textit{Construction procedure for equipment}.$$

To understand experimental computation both must be analysed.

6. Concluding remarks and further work

Experimental computation is not well understood even in the case of kinematics, possibly the simplest physical theory. We have discussed a methodology, based on four principles, and studied a simple example of hyper-computation in Newtonian Kinematics. In this process we sought:
(i) simple kinematic sub-theories to isolate physical assumptions about examples;
(ii) systems that are hyper-computers;
(iii) transparent embeddings so we can take apart and inspect the border between computable and non-computable; and
(iv) necessary and sufficient *physical* conditions for the sets and functions to become computable and the systems to become computers.

Clearly, the methodology can be applied to other examples, including those previously published in the literature. This is not easy and constitutes new work as many examples are insufficiently detailed as published, perhaps because they are complicated.

Problem 6.1. Apply the method to a large number of examples based on wide range of physical theories.

In Beggs and Tucker [3], we have discussed its application to some systems of Newtonian kinematics (e.g., Smith [34]) and of relativistic hyper-computation (e.g., [15,21,11]). Basic physics has been used in many discussions of computation and hyper-computation such as Moore [20], da Costa and Doria [7,8], Reif et al. [29,30], and Kieu [16]. The work of Smith [34] is close in spirit to the framework described in Section 2: he has a set of theories in which he studies the n-body systems and addresses the issues of Principle 3 above. Here the infinite computational resource comes from the arbitrarily high velocities attained in non-collision singularities in the n-body system, and the study of "topologically" different trajectories which can be used to achieve this. One might also consider complexity issues, e.g. Yao [48].

In describing the methodology, we have already drawn attention (in Section 2.2) to potential problems to do with axiomatisation and meta-theory. This is a long term task:

Problem 6.2. Guided by applications of the method to examples, examine the axiomatic, deductive and semantic properties of physical theories, and how they are refined and combined.

We have used the marble run to motivate the use of languages to define the operations to be used in experimental procedures and the construction of equipment. Working with classical logic gives us the freedom to postulate such exotic examples and worry about the consequences afterwards. Constructive logic demands a more operational and finite analysis of mathematical ideas. How does it cope with physical theories?

Problem 6.3. Analyse experimental computation by examples, such as the marble run, by considering the physical theories from a constructive point of view.

Let us note some further mathematical work connected to the specific example here. We have shown that for each set $A \subseteq \mathbb{N}$ there exists a valid Newtonian kinematic system M_A, bounded by a 3-dimensional box, that can decide the membership of the subset A, operating in a fixed finite time interval and using a fixed finite amount of energy for all inputs. In the light of the theorem, an open technical problem about systems is this:

Problem 6.4. For all valid Newtonian kinematic systems that possess both lower and upper bounds on space, time, mass, velocity and energy, are the sets and functions computable by experiment also computable by algorithms?

We conjecture that the answer is "Yes". To prove this, one needs to study general classes of experimental procedures and equipment.

In Newtonian mechanics we are allowed to shrink space and accelerate time. Of course, a mechanical system that exploits the infinite divisibility of space, and uses arbitrarily small components with no *lower* bounds on units of space, is not a model of atomic theory. But such examples, properly analysed, give insight into the physical foundations of computability. Plenty of examples are needed and the precise *physical* concepts, laws and conditions identified that permit or prevent non-computable functions and behaviours.

Our idea of analysing experimental computation using experimental procedures and constructible equipment and programming languages for defining them seems to be new. It advances theoretically the informal reflections of Geroch and Hartle [12] on the computability of measurements in physical experiments. Our examples show that the notion of constructible equipment in Newtonian mechanics must be analysed in great detail and that this is complicated. Roughly speaking, we seek a formal theory of Gedanken experiments capable of underpinning computability as follows:

Problem 6.5. Extend theoretical mechanics by a mathematical theory of experimental computation including languages for experimental procedures and construction procedures for equipment, and show that
(i) the sets and functions computable by experiment are precisely the same as, or equivalent to, those computable by algorithms;
(ii) there are sets that can be decided in polynomially bounded space and time by experimental computation with mechanical systems but cannot be decided by algorithms in polynomial space and time?

One goal of this direction of research, from physical theory to computability, is, roughly speaking, to explore rigorously and systematically the problem:

Problem 6.6. To derive forms of the Church–Turing thesis as laws in physical theories.

References

[1] W. Balzer, C.U. Moulines, J.D. Sneed, An Architectonic for Science, D Reidel/Kluwer, 1987.
[2] E.J. Beggs, J.V. Tucker, Computations via experiments with kinematic systems, Research Report 4.04, Department of Mathematics, University of Wales Swansea, March 2004 or Technical Report 5-2004, Department of Computer Science, University of Wales Swansea, March 2004.
[3] E.J. Beggs, J.V. Tucker, Embedding infinitely parallel computation in Newtonian kinematics, Applied Mathematics and Computation 178 (2006) 25–43.
[4] M.A. Bunge, Philosophy of Physics, D Reidel, 1973.
[5] S.B. Cooper, P. Odifreddi, Incomputability in nature, in: S.B. Cooper, S.S. Goncharov (Eds.), Computability and Models: Perspectives East and West, Kluwer Academic/Plenum, Dordrecht, 2003, pp. 137–160.
[6] H.C. Corben, P. Stehle, Classical Mechanics, 2nd edn, Dover, 1994.
[7] N.C.A. da Costa, F.A. Doria, Undecidability and incompleteness in classical mechanics, International Journal of Theoretical Physics 30 (1991) 1041–1073.
[8] N.C.A. da Costa, F.A. Doria, Classical physics and Penrose's thesis, Foundations of Physics Letters 4 (1991) 363–373.
[9] E.B. Davies, Building infinite machines, British Journal for the Philosophy of Science 52 (2001) 671–682.
[10] M. Davis, The myth of hypercomputation, in: C. Teuscher (Ed.), Alan Turing: Life and Legacy of a Great Thinker, Springer, 2004, pp. 195–211.
[11] J. Durand-Lose, Abstract geometric computation for Black hole computation, in: M. Margenstern (Ed.), Universal Turing Machines and Computations, in: Springer Lecture Notes in Computer Science, vol. 3354, 2005, pp. 175–186.
[12] R. Geroch, J.B. Hartle, Computability and physical theories, Foundations of Physics 16 (1986) 533–550.
[13] D.S. Graça, J.F. Costa, Analog computers and recursive functions over the reals, Journal of Complexity 19 (5) (2003) 644–664.
[14] A.V. Holden, J.V. Tucker, H. Zhang, M. Poole, Coupled map lattices as computational systems, American Institute of Physics — Chaos 2 (1992) 367–376.
[15] M. Hogarth, Predictability, computability and spacetime, Ph.D. Thesis, Cambridge University, 2002.
[16] T.D. Kieu, Quantum algorithm for Hilbert's tenth problem, International Journal of Theoretical Physics 42 (7) (2003) 1461–1478.
[17] G. Kreisel, A notion of mechanistic theory, Synthese 29 (1974) 9–24.
[18] L. Lipshitz, L.A. Rubel, A differentially algebraic replacement theorem, Proceedings of the American Mathematical Society 99 (2) (1987) 367–372.
[19] G. Ludwig, Die Grundstrukturen Einer Physikalischen Theorie, 2nd edn, Springer, 1990.
[20] C. Moore, Unpredictability and undecidability in dynamical systems, Physical Review Letters 64 (1990) 2354–2357.
[21] I. Nemeti, Gy. David, Relativistic computers and the Turing barrier, Applied Mathematics and Computation 178 (2006) 118–142.
[22] M.A. Nielsen, I.L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, 2000.
[23] T. Ord, Hypercomputation: Computing more than the Turing machine, Applied Mathematics and Computation 178 (2006) 143–153.
[24] R. Penrose, The Emperor's New Mind, Oxford University Press, 1989.
[25] M.B. Pour-El, Abstract computability and its relations to the general purpose analog computer, Transactions of the American Mathematical Society 199 (1974) 1–28.
[26] M.B. Pour-El, J.I. Richards, A computable ordinary differential equation which possesses no computable solution, Annals of Mathematical Logic 17 (1979) 61–90.
[27] M.B. Pour-El, J.I. Richards, The wave equation with computable initial data such that its unique solution is not computable, Advances in Mathematics 39 (1981) 215–239.
[28] M.B. Pour-El, J.I. Richards, Computability in Analysis and Physics, in: Perspectives in Mathematical Logic, Springer-Verlag, Berlin, 1989.
[29] J.H. Reif, J.D. Tygar, A. Yoshida, Computability and complexity of ray tracing, Discrete and Computational Geometry 11 (3) (1994) 265–287.
[30] J.H. Reif, S.R. Tate, The complexity of N-body simulations, in: Automata, Languages and Programming, 20th Proceedings, August 1993, in: Springer Lecture Notes, vol. 700, 1993, pp. 162–176.
[31] C. Shannon, Mathematical theory of the differential analyser, Journal of Mathematics and Physics 20 (1941) 337–354.
[32] J. Schroeter, Zur Matatheorie der Physik, De Gruyter, 1996.
[33] H.T. Siegelmann, Neural Networks and Analog Computation: Beyond the Turing Limit, Birkhäuser, Boston, 1999.
[34] W.D. Smith, Church's thesis meets the N-body problem, Applied Mathematics and Computation 178 (2006) 154–183.
[35] V. Stoltenberg-Hansen, J.V. Tucker, Concrete models of computation for topological algebras, Theoretical Computer Science 219 (1999) 347–378.
[36] V. Stoltenberg-Hansen, J.V. Tucker, Computable and continuous partial homomorphisms on metric partial algebras, Bulletin for Symbolic Logic 9 (2003) 299–334.
[37] W. Stegmueller, The Structure and Dynamics of Theories, Springer, 1976.
[38] J.V. Tucker, J.I. Zucker, Computable functions and semicomputable sets on many sorted algebras, in: S. Abramsky, D. Gabbay, T. Maibaum (Eds.), in: Handbook of Logic for Computer Science, vol. V, Oxford University Press, 2000, pp. 317–523.
[39] J.V. Tucker, J.I. Zucker, Abstract versus concrete computation on metric partial algebras, ACM Transactions on Computational Logic 5 (4) (2004) 611–668.
[40] J.V. Tucker, J.I. Zucker, Computable total functions on metric algebras, universal algebraic specifications and dynamical systems, Journal of Algebraic and Logic Programming 62 (2005) 71–108.
[41] J.V. Tucker, J.I. Zucker, A network model of analogue computation over metric algebras, in: S.B. Cooper, B. Lwe, L. Torenvliet (Eds.), New Computational Paradigms, Proceedings of Computability in Europe, June 2005, Amsterdam, in: Springer Lecture Notes, vol. 3526, 2005, pp. 515–529.

[42] K. Weihrauch, Computable Analysis, An Introduction, Springer-Verlag, Heidelberg, 2000.
[43] K. Weihrauch, N. Zhong, Is wave propagation computable or can wave computers beat the Turing machine? Proceedings of London Mathematical Society 85 (2002) 312–332.
[44] K. Weihrauch, N. Zhong, Is the Schrödinger propagator Turing computable? in: J.E. Blanck, V. Brattka, P. Hertling (Eds.), Computability and complexity in Analysis, in: Springer Lecture Notes in Computer Science, vol. 2064, 2001.
[45] C.-F. von Weizsäcker, Die Einheit der Natur, Hanser Verlag, 1971–1995.
[46] C.-F. von Weizsäcker, Die philosophische Interpretation der modernen Physik, Nova Acta Leopoldina/Abhandlungen der deutschen Akademie der Naturforscher Leopoldina 37 (2) (1973) No. 207.
[47] S. Wolfram, A New Kind of Science, Wolfram Media, Champaign, 2002.
[48] A. Yao, Classical physics and the Church Turing thesis, Journal ACM 50 (2003) 100–105.
[49] M. Ziegler, Computational power of infinite quantum parallelism, International Journal of Theoretical Physics 44 (2005) 2057–2071.