Newtonian systems, bounded in space, time, mass and energy can compute all functions[1]

E.J. Beggs[2] and J.V. Tucker[3]

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, especially on the issue of the existence of hyper-computers. First, we present some principles for making a theoretical analysis of computation by physical systems. We focus on the role of *simple* examples that are models of *simple* subtheories of physical theories. Secondly, we present such a simple example for Newtonian Kinematics and explore its meaning.

We prove 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 theorem implies that *simple Newtonian mechanical systems that are bounded in space, time, mass and energy can compute all possible functions on discrete data.* The system is a form of marble run and is a model of a small fragment of Newtonian Kinematics.

Then, we use the example to reflect on the nature of computation by physical systems. The proof of the theorem shows how *any* information (coded by some A) can be embedded in the static structure of simple kinematic equipment and retrieved by a simple experimental procedure. This suggests that the idea of equipment is problematic when computing under Newtonian mechanics. The example shows that a formal theory for experimental computation needs restrictive conditions on the notion of experimental procedure and the construction of equipment. We conjecture that using such a restricted mechanics the functions computed by experimental computation are "equivalent" to those computed by algorithms, i.e. the partial computable functions.

---

[2]Department of Mathematics. Email: e.j.beggs@swansea.ac.uk
[3]Department of Computer Science. Email: j.v.tucker@swansea.ac.uk

1 Introduction

The extraordinary development of the theory of computation since the 1930s has been based on its mathematical abstraction from the physical world. In consequence, the rise of digital computation has granted Computer Science its intellectual independence from Physics and Electronics. The maturity of our mathematical theories of computation allow us to look at their physical foundations in new ways. Research is prompted by a need to understand interfaces between algorithms and physical technologies (e.g., in new the problems of quantum information processing, or old problems of analogue computers), or by a fundamental curiousity about the information processing in physical systems.

What is the physical basis of computation? What is the physical theory of the 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 we may think of as a part of nature, a machine, or a combination of both, that performs the computation, and of

(b) a physical theory, which we use to reason about the system's behaviour and the computation.

The concept can be found in modelling physical and biological systems, e.g., in classical wave mechanics [18, 19, 30, 31, 24, 28] or in the emergent behaviour of cellular lattices and neural networks [10, 21, 32]. It can also be found in technologies for designing machines, e.g., analogue computers ([20, 16, 12, 9] or quantum computers ([14]). We have classified and surveyed the literature in our [1]. 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 will argue that, in contrast, the idea of experimental computation is not understood. At its heart are the concepts of *experimental procedure*, which is closely related to

that of algorithmic procedure, and *experimental equipment*, both of which are dependent upon a physical theories. Both 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 [11], which have stimulated research in Computable Analysis. Over the years, there have been physical examples (e.g., [13]), undecidability results for classes of dynamical systems (e.g., [17, **?**]), philosophical speculations on experimental computation (e.g., [15, 5, 3, 7, 33]). 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 [1]).

In this paper we will examine computation by idealised experiments with idealised physical systems. First, we propose some principles for making a theoretical analysis of computation by physical systems. The problem we address 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 subtheories of physical theories.* Simplicity is needed to put all the concepts, structures and techniques involved in the process under a theoretical microscope. Embedding sets of naturals into kinematic systems is one source of such examples.

Secondly, we present a simple system that is a model for Newtonian Kinematics and analyse its implications. Now, in our [1], 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 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 not used on the bagatelles. Our first task is to 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 theorem shows that non-computable finite bounded physical systems exist in a dramatic way: 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?* Thus, our final task is to reflect on the concept of experimental computation in the light of the theorem. We consider the ideas of experimental procedure and equipment.

Now, the proof 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 in Newtonian mechanics.

To answer the questions above, we find that the idea of experimental computation involves much more than an experimental procedure applied to a black box containing an arbitrary given physical system. Crucially, it involves the

(i) application of a sequence of *experimental actions* to the marble run, and

(ii) *process* of *assembly* or *construction* of the equipment.

We propose a new approach of using programming languages to analyse both experimental procedures and construction procedures. Our theorem shows that a refinement of mechanics is needed in which to define classes of *constuctible equipment*, and 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 mechanics the functions computed by experimental computation are "equivalent" to those computed by algorithms, i.e. the partial computable functions.

The structure of the paper is this. In Section 2 we present four principles for an investigation of experimental computations. In Section 4 we describe the construction of a simple type of marble run that decides the membership relation for A under Newtonian Kinematics. In Section 5 we reflect on the examples and the nature of experimental procedures and equipment. Finally, in Section 6, we discuss some open problems.

This paper is a sequel to Beggs and Tucker [1], 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 [2]. Technically, only elementary Newtonian mechanics is needed to follow our arguments. However, some knowledege of the different approaches to the fomulation of the problem and its "solution" is necessary. Thus, as background, it is helpful if the reader is familiar with [1] and with the theory of the functions computable by algorithms on discrete data, and its extension to continuous data (see: [19, 23, 25, 26, 29]).

2 On the analysis of 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;

(iii) the operational procedures necessary to initialize, control and observe the course of the experiment in time;

(iv) calculations that might be involved in performing the experiment;

(v) the equipment of system and how it is to be designed, created and used.

In this Section we propose a set of basic principles for investigating experimental computation and hyper-computation by physical systems.

2.1   Role of Physical Theories

Physical theories play a complicated role in experimental computation. In seekng 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 that is the central object of theoretical study. Here we propose four principles and stages to an investigation of some type of experimental computation.

1. Defining a physical subtheory: *Define precisely a subtheory T of a physical theory and examine experimental computation by the systems that are valid models of the subtheory 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 subtheory of Newtonian Mechanics. By working on examples, conceptual precision and mathematical rigour will surely reduce the subtheory 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 Subtheories *In computing with a physical system C, specify the smallest subtheory or fragment T of the physical theory that is needed to construct, operate, reason about, and validate C.*

The chosen subtheory 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 subtheory 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.

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.*

Subtheories 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 subtheory T.*

Clearly, as with any embedding problem, we are interested in embedding *complex algorithmic computations* into *simple physical systems* that are models *simple physical subtheories*. For example, to answer the question do 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 [1]). 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 emdeddings 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 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 examples and need 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.*

3. Mapping the border between computer and hyper-computer in physical theory: *Analyse what properties of the subtheory 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 subtheories 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 [6]. Indeed, it is an interesting problem to investigate which models of computers can be represented or implemented in a physical theory T.

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 subtheory. Criticism of the system might require strengthening the subtheory T in different ways.*

We have emphasised the autonomy and independence of the physical theory since we take it and its subtheories 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 subtheory in different ways, perhaps adding more laws in the search of 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 subtheory 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.

3 Experimental Computation with 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, subtheories 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 subtheory 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.

Newtonian kinematics has been used in many discussions of computation and hyper-computation such as Moore [13], da Costa and Doria [4, 5], Yao [33]. The work of Smith [22] is particularly close in spirit to the framework described in Section 2: his portfolio of theories in which he studies the N-body systems 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.

In contrast to this, our simple example will need a simple fragment of Newtonian Kinematics. 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 will look at the subtheory in more detail after we have explained the system in the next section.

4   The Marble Run

Here is the proof of the 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 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

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

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 $R(n) = \pi/2^{n+7}$.

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 figure 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.
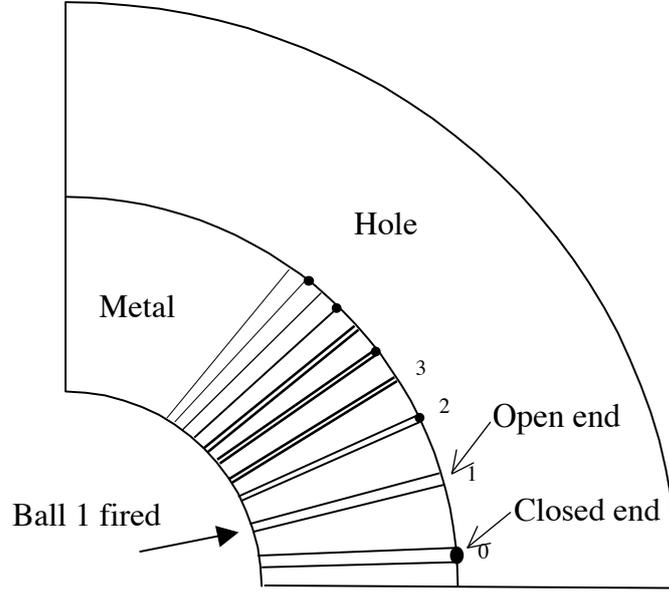

Figure 1: The Marble Run

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

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

10

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.

Variations of the marble run. If we work with the simpler assumption of point particles then the complications of selection can be dispensed with. For convenience we have assumed that there is no friction: it could be included with adjustments to the speeds of the particles.

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

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

5 Experimental procedures and equipment

We will consider the marble run using the framework discussed in Section 2. First let us check on the minimal subtheory.

5.1 The Minimal Subtheory and the Marble Run Procedure

What does the minimal subtheory for the marble run contain?

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) place a particle in space;

(i) project a particle in space;

(ii) observe the position of a particle in space;

(iii) measure time on a clock; and

There are also some parameters defining the mass and size of particles and their velocities, but, in the case of the marble run, the actions required are simple and uncontentious. Note that we can tolerate inaccuracies in measurements and that we did not calculate with even simple algebraic formulae. Clearly, for the procedure these actions determine the minimal kinematic subtheory T_M. To define this theory T_M fully and formally is a task we leave to another occasion. We can make T_M a little more precise as follows.

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

exp-pseudocode 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 become instructions and their combination by control constructs become experimental procedures. The language of the pseudocode is derived by T_M.

What does the equipment M_A contribute to the minimal theory T_M?

First we note that the marble run M_A satisfies the laws of T_M. The structure of the system 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 view it operationally and 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:

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 allow initialisation and observation of the equipment, which is otherwise considered to be a black box. The procedure schedules the actions, step by step, indexed by one or more clocks. Briefly,

Experimental computation = Experimental procedure + 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.

5.2.1 Experimental procedures

Inspired by experimental procedures expressed in pseudocode (Section 5.1), we have begun exploring the idea associating with a physical theory T languages for expressing experimental procedures. Each language $EP(T)$ 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,

Experimental procedure = Experimental actions + Calculations + Control and Scheduling.

Specifically, the actions or basic instructions of $EP(T)$ 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 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. It is a black box and is governed by physical laws. 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.

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?

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,

according to a blueprint that is a construction 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 wide range of primitive construction actions to consider. For example, the steps involving the geometry of the space or environment 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 S based on a theory T leads to a *construction language $CP(T)$* for expressing construction procedures. We define Newtonian equipment to be *constructible* in the construction language $CP(T)$ if it is definable by a construction procedure of $CP(T)$.

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

Clearly, the use of construction languages allow us to place conditions on the form of equipment and explain why the marble run might not be valid for 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 will comment on how a construction language is combined with the experimental procedure language to form a complete language for experimental computation and argue why such a language cannot express the marble run.

In a complete specification of an experimental computation there are two cases:

(i) *Complete Equipment Specification.* All the steps of the construction procedure are completed before starting the steps of an experimental procedure.

(ii) *Partial Equipment Specification.* The steps of the construction procedure are interleaved with steps of an experimental procedure.

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 defined by the instructions of the construction language - are finitely many.

For any equipment that involves an 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 (or semidecidable) by experiment with the marble run M_A if, and only if, A is decidable (or semidecidable) by experimental procedure from $EP(T_M)$ applied to M_A and M_A is partially $CP(T_M)$ constructible (or semiconstructible).*

(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 the working Definition 5.1. The pseudocode is a sketch proof that A is decidable by an experimental procedure from $EP(T_M)$. 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 $CP(T_M)$ 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 equipment specification language $CP(T)$ 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 $CP(T)$. Thus, under Definition 5.1, marble runs do *not* compute all functions on $\mathbb{N}$!

One idea is to suppose that A should be algorithmically decidable or semidecidable for use in $CP(T_M)$. There is an obvious argument that the idealised CAD tool that cuts the metal needs an algorithm for A in order to construct M_A. If we do introduce classical computability concepts into $CP(T_M)$ 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.

To introduce computability into the physical theory via $CP(T_M)$ run risks of circularity for we are seeking to understand the physical foundations of computability theory. So the conjecture remains and it is a subtle problem to formalise the languages $CP(T_M)$ and $EP(T_M)$ and show or refute this.

6   Concluding remarks

Experimental computation is not well understood even in the case of kinematics, possibly the simplest physical theory. We have discussed a framework, based on four principles, and studied a simple example of hyper-computation in Newtonian Kinematics. In this process we sought:
(i) a simple kinematic subtheories 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.

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 6.1. *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 use 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. For definitive answers, 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 [8] 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 is this complicated. Roughly speaking, we seek a theory of Gedanken experiments capable of underpinning computability as follows:

Problem 6.2. *Extend theoretical mechanics by a mathematical theory of experimental computation including languages for experimental procedures and the 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 computablity, is, roughly speaking, to explore the problem:

Problem 6.3. *To derive forms of the Church-Turing Thesis as laws in physical theories.*

References

[1] E J Beggs and 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.

[2] E J Beggs and J V Tucker, Embedding infinitely parallel computation in Newtonian kinematics, Research Report ?.05, Department of Mathematics, University of Wales Swansea, March 2005 or Technical Report ?-2005, Department of Computer Science, University of Wales Swansea, March 2005.

[3] S B Cooper and P Odifreddi, Incomputability in nature, in S. B. Cooper and S. S. Goncharov, (eds.), *Computability and Models: Perspectives East and West*, Kluwer Academic/Plenum Publishers, Dordrecht, 2003, pp. 137-160.

[4] N C A da Costa and F A Doria, Undecidability and incompleteness in classical mechanics, *International Journal of Theoretical Physics* 30 (1991), 1041-1073.

[5] N C A da Costa and F A Doria, Classical physics and Penrose's thesis, *Foundations of Physics Letters* 4 (1991), 363-373.

[6] E B Davies, Building infinite machines,

[7] M Davis, The myth of hypercomputation, in *Turing Festschrift*, Springer, in preparation.

[8] R Geroch and J B Hartle, Computability and physical theories, *Foundations of Physics* 16 (1986), 533-550.

[9] D S Graça and J F Costa, Analog computers and recursive functions over the reals, 2003, submitted.

[10] A V Holden, J V Tucker, H Zhang and M Poole, Coupled map lattices as computational systems, *American Institute of Physics - Chaos* 2 (1992), 367-376.

[11] G Kreisel, A notion of mechanistic theory, *Synthese* 29 (1974), 9-24.

[12] L Lipshitz and L A Rubel, A differentially algebraic replacement theorem, *Proceedings of the American Mathematical Society* 99(2) (1987), 367 – 372.

[13] C Moore, Unpredictability and undecidability in dynamical systems, *Physical Review Letters* 64 (1990), 2354 – 2357.

[14] M A Nielsen and I L Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, 2000.

[15] R Penrose, *The Emperor's New Mind*, Oxford University Press, 1989.

[16] 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.

[17] M B Pour-El and J. I. Richards, A computable ordinary differential equation which possesses no computable solution, *Annals of Mathematical Logic* 17 (1979), 61 – 90.

[18] M B Pour-El and 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.

[19] M B Pour-El and J. I. Richards, *Computability in Analysis and Physics*, Perspectives in Mathematical Logic, Springer-Verlag, Berlin, 1989.

[20] C Shannon, Mathematical theory of the differential analyser, *Journal of Mathematics and Physics* 20 (1941), 337-354.

[21] H T Siegelmann, *Neural networks and analog computation: Beyond the Turing limit*, Birkhäuser, Boston, 1999.

[22] W D Smith, Church's thesis meets the N-body problem, in *Applied Mathematics and Computation*, to appear.

[23] V Stoltenberg-Hansen and J V Tucker, Concrete models of computation for topological algebras, *Theoretical Computer Science* 219 (1999), 347 – 378.

[24] V Stoltenberg-Hansen and J V Tucker, Computable and continuous partial homomorphisms on metric partial algebras, *Bulletin for Symbolic Logic* 9 (2003), 299 – 334.

[25] J V Tucker and J I Zucker, Computable functions and semicomputable sets on many sorted algebras, in S. Abramsky, D. Gabbay and T Maibaum (eds.), *Handbook of Logic for Computer Science* volume V, Oxford University Press, 2000, 317 – 523.

[26] J V Tucker and J I Zucker, Abstract versus concrete computation on metric partial algebras, *ACM Transactions on Computational Logic*, 5 (4) (2004) 611-668.

[27] , J V Tucker and 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.

[28] J V Tucker and 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, Amsterdam June 2005*, Springer Lecture Notes 3526, 515-529, 2005.

[29] K Weihrauch, *Computable Analysis, An introduction*, Springer-Verlag, Heidelberg, 2000.

[30] K Weihrauch and N Zhong, Is wave propogation computable or can wave computers beat the Turing machine?, *Proceedings of London Mathematical Society* 85 (2002), 312-332.

[31] K Weihrauch and N Zhong, Is the Schrödinger propagator Turing computable?, in J. E. Blanck, V. Brattka and P. Hertling (eds.), *Computability and complexity in Analysis*, Springer Lecture Notes in Computer Science, Volume 2064, 2001.

[32] S Wolfram, *A New Kind of Science*, Wolfram Media, Champaign, 2002.

[33] A Yao, Classical physics and the Church Turing thesis, *Journal ACM* 50 (2003), 100-105.