Tracking computability of GPAC-generable functions

DIOGO POÇAS, *LASIGE, Departamento de Informática, Faculdade de Ciências, Universidade de Lisboa, 1749-016 Lisbon, Portugal.*
E-mail: dmpocas@fc.ul.pt

JEFFERY ZUCKER, *Department of Computing and Software, McMaster University, Hamilton, ON, L8S4K1, Canada.*

Abstract

Analog computation attempts to capture any type of computation, that can be realized by any type of physical system or physical process, including but not limited to computation over continuous measurable quantities. A pioneering model is the General Purpose Analog Computer (GPAC), initially presented by Shannon in 1941. The GPAC is capable of manipulating real-valued data streams; however, it has been shown to be strictly less powerful than other models of computation on the reals, such as computable analysis.

In previous work, we proposed an extension of the Shannon GPAC, denoted LGPAC, designed to overcome its limitations. Not only is the LGPAC model capable of expressing computation over general data spaces $\mathcal{X}$, but it also directly incorporates approximating computations by means of a limit module. An important feature of this work is the generalisation of the framework of the computation theory from Banach to Fréchet spaces.

In this paper, we compare the LGPAC with a digital model of computation based on effective representations (tracking computability). We establish general conditions under which LGPAC-generable functions are tracking computable.

Keywords: generalized computability, generalized recursion theory, computation on the reals, analog computation, Shannon GPAC, tracking computability

1 Introduction

A central goal in computability theory is to establish equivalences between disparate notions of computation; such equivalence results serve as strong indications of the validity of the theory as a whole, as they suggest robustness (or perhaps, indifference) against the choice of a particular model of computation.

In the framework of digital computation, such considerations have led to the celebrated Church–Turing thesis that asserts that any realizable method of computation has the same computational power as the Turing machine. However, the picture is not so clear in the case of computation over more general data spaces, or analog computation. Analog computation, as conceived by Kelvin [21], Bush [2] and Hartree [6], is a form of experimental computation with physical systems called analog devices or analog computers. Historically, data are represented by measurable physical quantities, including lengths, shaft rotation, voltage, current, resistance, etc.

The General Purpose Analog Computer (GPAC) was introduced by Shannon [17] as a model of Bush's Differential Analyzer [2]. Shannon discovered that a function can be generated by a GPAC if, and only if, it is differentially algebraic. In particular, this implies that non-differentially algebraic functions, such as the gamma function, cannot be generated by the Shannon GPAC.

doi:10.1093/logcom/exaa081

In previous works [11, 13], we proposed different extensions of the Shannon GPAC, attempting to overcome its limitations. In particular, our models express computation over general data spaces $\mathcal{X}$ beyond real numbers and directly incorporate approximating computations by means of a *limit module*. The goal of this paper is to connect the LGPAC (GPAC + limits) with other such models of computation. Specifically, we shall consider the notion of *tracking computability* [19, 22], which we take as a paradigm for digital computation. The idea of tracking computability comes from Mal'cev [10], and it has been found to be equivalent (under reasonable conditions) to a number of other well-known digital computation models [19, 20, 22, 24]. In this work, we find suitable conditions that guarantee that a function generated by an LGPAC is also tracking computable.

We begin by introducing both notions of computability: for the GPAC model, we describe the channels, modules, input-output operator and fixed point semantics; for tracking computability, we study computable structures and effective representations. In the most technical part of the paper, we prove tracking computability of the functions associated with the LGPAC modules and of the input-output operator of an LGPAC. Finally, we attempt to prove tracking computability of LGPAC-generable functions; in order to achieve this, we assume an additional condition, which we call *effective well-posedness*.

This research is part of a project to compare the strengths of various models of analog and digital computation. In the present case (LGPAC and tracking computability) we have been successful in one direction, while the other direction remains an open problem. An important feature of this paper is the use of Fréchet spaces as a framework for computation theory on the reals. The significance of this is given in Remarks 2.1 and 3.1 below.

In regard to the original content of this paper, we remark that the paper [1] already shows an equivalence between a GPAC-like model for *real* computation (which includes approximability) and computable analysis (which is closely related to tracking computability; papers [22] and [23] have some equivalence results). However, our model differs from the model in that paper in two critical ways: computation on general data spaces is allowed, and approximability is directly incorporated by means of limit modules. Hence, the two models are not obviously comparable. The technical notion of effective well-posedness (Definition 6.2) is also an original idea.

2 Preliminaries

Our model of computation is built over a data space $\mathcal{X}$ which represents the space of possible data points. Typical spaces of interest are $\mathbb{R}$ (the real numbers), $C(\mathbb{R})$ (continuous real functions of one real variable), $C^1(\mathbb{R})$ (continuously differentiable real functions) and so on. The results in this paper will be stated for *separable Fréchet spaces*, which satisfy the following assumptions.[1]

1. $\mathcal{X}$ is a Fréchet space with respect to a family of pseudonorms $\|\cdot\|_n$ (indexed by $n \in \mathbb{N}$); in particular, it is equipped with the vector space operations of addition and scalar multiplication, as well as a zero element $0 \in \mathcal{X}$. We recall the pseudonorm axioms:

1a. each pseudonorm $\|\cdot\|_n : \mathcal{X} \to \mathbb{R}_{\geq 0}$ is positive semidefinite ($\|x\|_n \geq 0$), scalable ($\|rx\|_n = |r|\|x\|_n$ for $r \in \mathbb{R}$) and subadditive ($\|x+y\|_n \leq \|x\|_n + \|y\|_n$);

1b. the family of pseudonorms separates points: if $\|x-y\|_n = 0$ for all n, then $x = y$;

1c. $\mathcal{X}$ is complete, that is, Cauchy sequences are convergent: for any sequence $(x_m) \in \mathcal{X}^{\mathbb{N}}$, if $\lim_{s,t\to\infty} \|x_s - x_t\|_n = 0$ for all n, then there exists $x \in \mathcal{X}$ such that $\lim_{s\to\infty} \|x_s - x\|_n = 0$;

[1] A detailed exposition of Fréchet spaces can be found in [15, Ch. V].

1d. for convenience, we additionally assume without loss of generality that the pseudonorms are nondecreasing: if $n \leq m$ then $\|x\|_n \leq \|x\|_m$.
2. Finally, $\mathcal{X}$ is separable, i.e. it has a countable dense subset. We fix an enumeration $\alpha_{\mathcal{X}} : \mathbb{N} \to \mathcal{X}_c$ of a countable dense subset $\mathcal{X}_c = \alpha_{\mathcal{X}}(\mathbb{N}) \subseteq \mathcal{X}$. We also assume for convenience that $\alpha_{\mathcal{X}}(0) = 0$.

REMARK 2.1
(**The use of Fréchet spaces**).
An important aspect of the present work lies in the *generalisation* of the framework for the study of computation theory on the reals, from Banach spaces [7, 13, 18, 20, 22] to Fréchet spaces.[2] The significance of this lies in the fact that some well-known models of analog computation, such as $C(\mathbb{R})$ or $C^n(\mathbb{T}, C(\mathbb{R}))$ $(n = 0, 1, 2, \ldots)$, are not Banach spaces but Fréchet spaces (see also Remark 3.1).

Note that the family of pseudonorms on a Fréchet space induces a metric as follows.

DEFINITION 2.2
(**Metric from pseudonorms**).
Let $\mathcal{X}$ be a Fréchet space with pseudonorms $\|\cdot\|_n$, $n \in \mathbb{N}$. We define the *metric*

$$d(x, y) = \sup_{n \in \mathbb{N}} 2^{-n} \min\left(\|x - y\|_n, 1\right), \quad x, y \in \mathcal{X}. \tag{1}$$

PROPOSITION 2.3
(**Bounds on the pseudonorms and bounds on the metric**).
Let $\mathcal{X}$ be a Fréchet space with pseudonorms $\|\cdot\|_n$, $n \in \mathbb{N}$. Then for the metric defined by (1), the following hold for any $x, y \in \mathcal{X}$ and $n, m \in \mathbb{N}$:

$$\text{if } d(x, y) < 2^{-n-m}, \text{ then } \|x - y\|_n < 2^{-m};$$

$$\text{if } \|x - y\|_n < 2^{-m}, \text{ then } d(x, y) < 2^{-\min(n,m)}.$$

3 The LGPAC

We give a formal definition of the LGPAC model.[3] The main objects of our study are analog networks or analog systems, whose main components can be viewed as follows:

Analog network = data + time + channels + modules.

As already mentioned, we model data as elements of a separable Fréchet space $\mathcal{X}$. We will use the nonnegative real numbers as a continuous model of *time* $\mathbb{T} = [0, \infty)$. We consider two types of *channels*: *scalar channels* carry constant values in $\mathcal{X}$, whereas *stream channels* carry continuously differentiable streams in $C^1(\mathbb{T}, \mathcal{X})$.

REMARK 3.1
(**Generation of Fréchet spaces**).
If $\mathcal{X}$ is a Fréchet space, so is $C^1(\mathbb{T}, \mathcal{X})$; this however does not hold in general for Banach spaces, which again explains our use of Fréchet spaces (see Remark 2.1). In particular, we can define the

[2] Fréchet spaces were used in this connection in the first author's doctoral thesis [12] and papers based on it [11, 13] including the present paper.
[3] More details can be found in [11, 13].

following family of nondecreasing pseudonorms on $C^1(\mathbb{T}, \mathcal{X})$,[4]

$$\|u\|_n = \|u(0)\|_n + \sup_{0 \le t \le n} \|u'(t)\|_n. \tag{2}$$

This gives a metric (by Defintion 2.2) and hence a topology for $C^1(\mathbb{T}, \mathcal{X})$ and the spaces $\mathcal{I}$, $\mathcal{M}$, $\mathcal{O}$ used in Definition 3.5.

Note that many useful properties of integration can be extended to $C^1(\mathbb{T}, \mathcal{X})$; in particular, for $u \in C^1(\mathbb{T}, \mathcal{X})$, $\nu \in \mathbb{N}$ and $t \in [0, \nu]$, we have the bound (which we will use later)

$$\|u(t)\|_\nu \le \|u(0)\|_\nu + \int_0^t \|u'(s)\|_\nu \mathrm{d}s \le \|u(0)\|_\nu + t \sup_{t \le \nu} \|u'(t)\|_\nu \le \nu \|u\|_\nu. \tag{3}$$

Each *module M* has zero, one or more input channels, and must have a single output channel; thus, it can be specified by a (possibly partially defined) *stream function*

$$F_M : A_1 \times \ldots \times A_k \rightharpoonup A_{k+1} \quad (k \ge 0),$$

where each of A_i, $i = 1 \ldots k+1$ is either $\mathcal{X}_i$ or $C^1(\mathbb{T}, \mathcal{X}_i)$ for some data space $\mathcal{X}_i$; and we use the symbol $\rightharpoonup$ to mean that F_M may be partial-valued. The Shannon GPAC is obtained if all $\mathcal{X}_i = \mathbb{R}$, and the following four types of modules are considered.

DEFINITION 3.2
(**Shannon modules**).
The *Shannon modules* are defined as follows:

- for each $c \in \mathbb{R}$, there is a *constant module* with zero inputs and one output $v(t) = c$;
- the *adder module* has two inputs u, v and one output w, given by $w(t) = u(t) + v(t)$;
- the *multiplier module* has two inputs u, v and one output w, given by $w(t) = u(t)v(t)$;
- the *integrator module* has a scalar input c (also called the initial setting), two stream inputs u, v and one output w, given by the Riemann–Stieltjes integral $w(t) = c + \int_0^t u(s)v'(s)\mathrm{d}s$.

We have previously extended the Shannon GPAC in two different ways.

1. General data spaces. In [13] we defined the $\mathcal{X}$-GPAC, allowing the study of functions of more than one variable. The main idea present in that paper is to extend the *output space*, that is, replacing $C^1(\mathbb{T}, \mathbb{R})$ with $C^1(\mathbb{T}, \mathcal{X})$, where $\mathcal{X}$ is a metric vector space. For example, we can think of $\mathcal{X}$ as the space of continuous real-valued functions on $\mathbb{R}^n$, that is, $\mathcal{X} = C(\mathbb{R}^n, \mathbb{R})$. In this way, our channels will now carry $\mathcal{X}$-valued streams of data $u : \mathbb{T} \to \mathcal{X}$, which correspond to functions of $n+1$ real variables, under the 'uncurrying' $\mathbb{T} \to (\mathbb{R}^n \to \mathbb{R}) \simeq \mathbb{T} \times \mathbb{R}^n \to \mathbb{R}$. Evidently, one of the independent variables, namely 'time', plays a different role from the others—it can be used as a variable for integration and taking limits.

This leads us to consider a multityped GPAC, which means that different channels may carry values over different data spaces. In particular, we shall fix one separable Fréchet space $\mathcal{X}$ and allow four channel types: $\mathbb{R}$-*variables*, $\mathcal{X}$-*variables*, $\mathbb{R}$-*streams* and $\mathcal{X}$-*streams*, which carry values in $\mathbb{R}$, $\mathcal{X}$, $C^1(\mathbb{T}, \mathbb{R})$ and $C^1(\mathbb{T}, \mathcal{X})$, respectively.

[4] Here, assumption (1d), that the original family of pseudonorms on $\mathcal{X}$ is nondecreasing, is required; alternatively, one could introduce a double-indexing family such as $\|u\|_{n,m} = \|u(0)\|_n + \sup_{0 \le t \le m} \|u'(t)\|_n$.

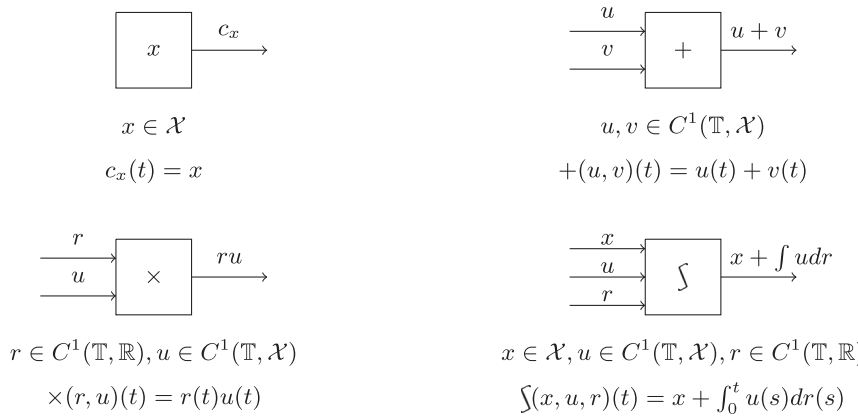

FIGURE 1. The four basic modules.

We generalize the Shannon modules to $C^1(\mathbb{T}, \mathcal{X})$, obtaining the four basic modules depicted in Fig 1 as box diagrams.[5] We also introduce the symbol '$\int$' to denote the operator associated with the integrator module; we can then write $\int(c, u, r) = c + \int u \mathrm{d}r$.

Observe that we are using *scalar* multiplication (of type $\mathbb{R} \times \mathcal{X} \to \mathcal{X}$) as the basis for the multiplication and integrator modules. At this level of generality, we cannot use multiplication (of type $\mathcal{X} \times \mathcal{X} \to \mathcal{X}$) because such an operation does not arise from the Fréchet space axioms. One can extend our model, assuming $\mathcal{X}$ is equipped with a multiplication operator, under suitable additional assumptions: of course, multiplication should be bilinear (i.e. distributive with respect to addition, and compatible with scalar multiplication), but more importantly, it should be *bounded* (e.g. one could assume $\|u \times v\|_\nu \leq \|u\|_\nu \|v\|_\nu$ for each pseudonorm ν). In Section 7, we will consider the case $\mathcal{X} = C(\mathbb{R})$ and extend the GPAC with a function multiplication module.

2. Limit modules. We introduced in [11] a *limit module* in order to incorporate approximating computations by means of *effective convergence*. If $M : \mathbb{N} \to \mathbb{N}$ is nondecreasing and $(g_n) \in \mathcal{X}^{\mathbb{N}}$, we say that (g_n) is an *M-convergent Cauchy sequence* if for all $\nu \in \mathbb{N}$ and $m, n \geq M(\nu)$ one has $d(g_m, g_n) < 2^{-\nu}$. Similarly, if $T \in C^1(\mathbb{T}, \mathbb{R})$ is nondecreasing and $u \in C^1(\mathbb{T}, \mathcal{X})$, we say that u is a *T-convergent Cauchy stream* if for all $\tau \in \mathbb{T}$ and $s, t \geq T(\tau)$ one has $d(u(s), u(t)) < 2^{-\tau}$.

We call such a non-decreasing function M (resp. T) a *discrete* (resp. *continuous*) *modulus of convergence*. A typical example is the identity function, either discrete (id : $\mathbb{N} \to \mathbb{N}$) or continuous (id $\in C^1(\mathbb{T}, \mathbb{R})$). We note that any M-convergent Cauchy sequence may be replaced by an id-convergent Cauchy sequence via a composition with its modulus of convergence. Similarly, a T-convergent Cauchy stream may be replaced by an id-convergent Cauchy stream. This brings us to the notion of a *limit operator*.

[5] By assumption, addition and scalar multiplication are defined on $\mathcal{X}$. The integral can be generalized to $C^1(\mathbb{T}, \mathcal{X})$ via Riemann sums: see, e.g. [16, p.89].

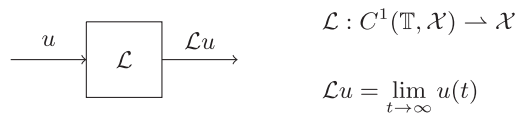

FIGURE 2. The limit module.

DEFINITION 3.3
(**Limit modules**).
For the data type $\mathcal{X}$, there is a *continuous limit module* with one input of type $C^1(\mathbb{T}, \mathcal{X})$ and one output of type $\mathcal{X}$. For input u, it outputs the id-convergent limit $\lim_{t\to\infty} u(t)$ (if it exists).

Observe that the limit module defines a partial-valued operator; it is only defined for those functions in $C^1(\mathbb{T}, \mathcal{X})$ that have an id-convergent limit.

DEFINITION 3.4
(**LGPAC**).
Let $\mathcal{X}$ be a separable Fréchet space. A *limit general purpose analog computer* (LGPAC) is a network built with $\mathbb{R}$-channels, $\mathcal{X}$-channels (carrying either constants or streams), the basic modules (constants, adders, multipliers, integrators) and the continuous limit module. Moreover, the channels connect the inputs and outputs of the modules, with the following restrictions: the only connections allowed are between an output and an input; each input may be connected to either zero or one output.

Thus, a GPAC channel may appear as an unconnected input (*proper input*), unconnected output (*proper output*), or connect an input with an output (*mixed input/output*).

DEFINITION 3.5
(**LGPAC semantics**, [13]).
Any LGPAC $\mathcal{G}$ induces an *input–output operator* $\Phi : \mathcal{I} \times \mathcal{M} \rightharpoonup \mathcal{M} \times \mathcal{O}$, where $\mathcal{I}$, $\mathcal{M}$, $\mathcal{O}$ denote the spaces of proper input, mixed input/output and proper output channels, respectively;

1. for variables $\mathbf{u}^I \in \mathcal{I}$, $\mathbf{u}^M \in \mathcal{M}$, $\mathbf{u}^O \in \mathcal{O}$, the *fixed point equation* is given by

$$\Phi(\mathbf{u}^I, \mathbf{u}^M) = (\mathbf{u}^M, \mathbf{u}^O); \tag{4}$$

2. $\mathcal{G}$ is *well-posed* on an open subset $U \subseteq \mathcal{I}$ if for all $\mathbf{u}^I \in U$ there is a unique $(\mathbf{u}^M, \mathbf{u}^O)$ such that (4) holds; and moreover, the solution map $\mathbf{u}^I \mapsto (\mathbf{u}^M, \mathbf{u}^O)$ describes a continuous function $F : U \to \mathcal{M} \times \mathcal{O}$ with domain U; we further say that $\mathcal{G}$ *generates* F, or that F is *LGPAC-generable*.

Although Definitions 3.4 and 3.5 refer to networks built with the LGPAC modules, it is not hard to see how they generalize to any choice of arbitrary modules, which would define a more abstract notion of *multityped GPAC*. Some of our results (namely Lemma 5.1) can be stated in this more general form.

4 Tracking computability

The procedure for defining tracking computability in general spaces has been extensively documented by many authors (see, e.g. [19, 20, 23, 24]). The basic construction consists of taking an enumeration of a countable dense subset, defining *computable* elements as those given by effective Cauchy sequences, and considering *tracking functions*. We assume that we have fixed an enumeration $\alpha_{\mathcal{X}} : \mathbb{N} \to \mathcal{X}_c$ of a (countable) dense subset $\mathcal{X}_c \subseteq \mathcal{X}$.

Let us also fix a family of computable bijections $\langle \cdot, \cdots, \cdot \rangle : \mathbb{N}^k \to \mathbb{N}$, for $k \in \mathbb{N}^+$ (say, the Cantor pairing function $\langle \cdot, \cdot \rangle$ for $k = 2$ and its generalizations to higher dimensions), as well as an enumeration $\{\cdot\} : \mathbb{N} \to (\mathbb{N} \rightharpoonup \mathbb{N})$ of the recursive functions (say, for $T \in \mathbb{N}$ the encoding of a one-input, one-output Turing machine, $\{T\}$ is the corresponding recursive function).

DEFINITION 4.1
(**Computability structure**).
Let $\mathcal{X}$ be a complete metric space and $(\mathcal{X}_c, \alpha)$ an enumerated countable dense subset. A *computability structure* $(\Omega_{\bar{\alpha}}, C_{\bar{\alpha}}, \bar{\alpha})$ is defined as follows.

1. The set of *valid codes*, $\Omega_{\bar{\alpha}}$, is the subset of $\mathbb{N}$ given by encodings of pairs of numbers $c = \langle T, M \rangle$ such that T is the index for a total recursive function $\{T\}$, M is the index for a total recursive discrete modulus of convergence $\{M\}$ and $(\alpha\{T\}(n))$ is an $\{M\}$-convergent Cauchy sequence.[6]
2. The *partial enumeration* $\bar{\alpha} : \mathbb{N} \rightharpoonup \mathcal{X}$ is the function with domain $\Omega_{\bar{\alpha}}$ such that for any $c = \langle T, M \rangle \in \Omega_{\bar{\alpha}}$, $\bar{\alpha}(c) = \lim_{n\to\infty} \alpha\{T\}(n)$.
3. The set of *computable elements* $C_{\bar{\alpha}} \subseteq \mathcal{X}$ is the range of $\bar{\alpha}$, i.e. $C_{\bar{\alpha}} = \bar{\alpha}(\mathbb{N})$.

EXAMPLE 4.2
(**Computability on** $\mathbb{R}$).
To construct a computability structure on the space $\mathcal{X} = \mathbb{R}$, we can take $\mathcal{X}_c = \mathbb{Q}$, and $\alpha = \alpha_{\mathbb{R}}$ as any standard enumeration of the rationals. This gives the set $C_{\bar{\alpha}}$ of computable reals.

EXAMPLE 4.3
(**Computability on** $C(\mathbb{R})$).
We define a computability structure on $\mathcal{X} = C(\mathbb{R})$, which is a Fréchet space with pseudonorms $\|f\|_n = \sup_{-n\le x\le n} |f(x)|$. We take $\mathcal{X}_c$ to be a countable subset of piecewise linear rational functions, defined as follows. For each $N \in \mathbb{N}$ and each tuple $(p_{-N^2}, \ldots, p_{-1}, p_0, p_1, \ldots, p_{N^2})$ of $2N^2 + 1$ rational numbers, we can consider a function $f : \mathbb{R} \to \mathbb{R}$ such that $f(x) = p_{-N^2}$ for $x \le -N$, $f(x) = p_{N^2}$ for $x \ge N$; $f(j/N) = p_j$ for $j \in \{-N^2, \ldots, 0, \ldots, N^2\}$ and f is piecewise linear on each interval $[j/N, (j+1)/N]$ for $j \in \{-N^2, \ldots, 0, \ldots, N^2 - 1\}$.

In this way, the role of N is both to increase the 'window size' and decrease the 'step size' of our approximation (see Fig 3). By using the bijections of type $\mathbb{N}^2 \to \mathbb{N}$ and $\mathbb{N}^{2N^2+1} \to \mathbb{N}$, and the enumeration $\alpha_{\mathbb{R}}$ from the previous example, we can define an enumeration $\alpha_{\mathcal{X}} : \mathbb{N} \to \mathcal{X}_c$. Specifically, the enumeration is as follows: for $e = \langle N, \langle m_{-N^2}, \ldots, m_{N^2} \rangle \rangle$, we define $\alpha_{\mathcal{X}}(e)$ to be the stream u built from N and the tuple $(p_{-N^2}, \ldots, p_{N^2})$ where $p_j = \alpha_{\mathbb{R}}(m_j)$ for each

[6] For ease of notation we write $\alpha\{T\}(n)$ instead of $\alpha(\{T\}(n))$.

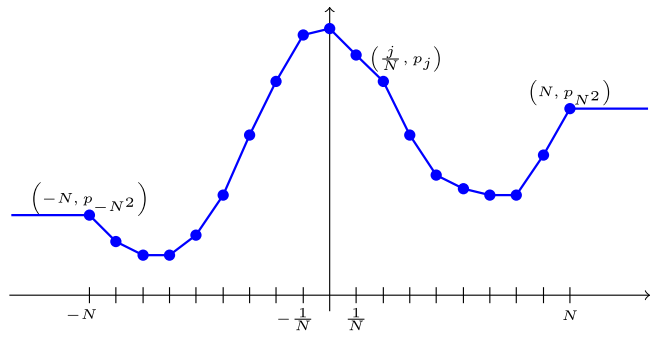

Figure 3. A piecewise linear rational function.

$j \in \{-N^2, \ldots, 0, \ldots, N^2\}$. Finally, we can apply the construction of Definition 4.1 and consider the set of computable elements $C_{\tilde{\alpha}}$. In this case, this set coincides with the familiar set of computable real functions, as seen in [14, 25], among others.

EXAMPLE 4.4
(**Computability on** $C^1(\mathbb{T}, \mathcal{X})$).
Given a computability structure on a separable Fréchet space $\mathcal{X}$, say with an enumeration $(\alpha_{\mathcal{X}}, \mathcal{X}_c)$, we shall construct a computability structure on the space of $\mathcal{X}$-streams $\mathcal{Z} = C^1(\mathbb{T}, \mathcal{X})$. We apply the same idea as in Example 4.3, but now we need to account for continuous differentiability. The idea is to construct an interpolant from a finite amount of 'data points'. If $u \in C^1(\mathbb{T}, \mathcal{X})$ then it has a derivative $u' \in C(\mathbb{T}, \mathcal{X})$. Therefore, we can approximate u' by a piecewise linear function and then integrate the approximation with respect to the time variable.

Formally, for each $N \in \mathbb{N}$ and each tuple $(x_0, y_0, \ldots, y_{N^2})$ of $N^2 + 2$ elements in $\mathcal{X}_c$, we consider the functions $u, v : \mathbb{T} \to \mathcal{X}$ such that: $v(t) = y_{N^2}$ for $t \geq N$; $v(j/N) = y_j$ for $j \in \{0, \ldots, N^2\}$; v is piecewise linear (as a function of t) and given by $v(t) = y_j + (y_{j+1} - y_j)(Nt - j)$ on each interval $[j/N, (j+1)/N]$, for $j \in \{0, \ldots, N^2 - 1\}$; finally, $u(t) = x_0 + \int_0^t v(s)\mathrm{d}s$.

By construction, each u is continuously differentiable and piecewise quadratic (Fig 4). Now let $\mathcal{Z}_c \subseteq \mathcal{Z}$ be the space of functions u considered above. Using the bijections of type $\mathbb{N}^2 \to \mathbb{N}$ and $\mathbb{N}^{N^2+2} \to \mathbb{N}$, and the enumeration $\alpha_{\mathcal{X}}$, we can define an enumeration $\alpha_{\mathcal{Z}} : \mathbb{N} \to \mathcal{Z}_c$. Specifically: for $e = \langle N, \langle m_0, m'_0, \ldots, m'_{N^2} \rangle \rangle$, we define $\alpha_{\mathcal{Z}}(e)$ to be the stream u built from N and the tuple $(x_0, y_0, \ldots, y_{N^2})$ where $x_0 = \alpha_{\mathcal{X}}(m_0)$ and $y_j = \alpha_{\mathcal{X}}(m'_j)$ for each $j = 0, \ldots N^2$.

$\mathcal{Z}_c$ is easily seen to be countable and dense in $\mathcal{Z}$. Thus, we can apply the construction of Definition 4.1 and obtain the computability structure $(\Omega_{\tilde{\alpha}_{\mathcal{Z}}}, C_{\tilde{\alpha}_{\mathcal{Z}}}, \bar{\alpha}_{\mathcal{Z}})$.

EXAMPLE 4.5
(**Computability on** $\mathcal{X} \times \mathcal{Y}$).
Given computability structures on spaces $\mathcal{X}, \mathcal{Y}$, one can define a computability structure on the product $\mathcal{X} \times \mathcal{Y}$ using the enumeration $\alpha_{\mathcal{X}\times\mathcal{Y}}(\langle \ell, r \rangle) = (\alpha_{\mathcal{X}}(\ell), \alpha_{\mathcal{Y}}(r))$. Note that pseudonorms (and a metric) on $\mathcal{X} \times \mathcal{Y}$ can be easily induced from $\mathcal{X}$ and $\mathcal{Y}$ as, e.g. $\|(x_1, y_1) - (x_2, y_2)\|_n = \|x_1 - x_2\|_n + \|y_1 - y_2\|_n$. It is also not hard to see that $C_{\tilde{\alpha}_{\mathcal{X}\times\mathcal{Y}}} = C_{\tilde{\alpha}_{\mathcal{X}}} \times C_{\tilde{\alpha}_{\mathcal{Y}}}$, and that this construction can be generalized to finite products of the form $\mathcal{X}_1 \times \ldots \times \mathcal{X}_N$ and (hence) to $\mathcal{X}^N$.

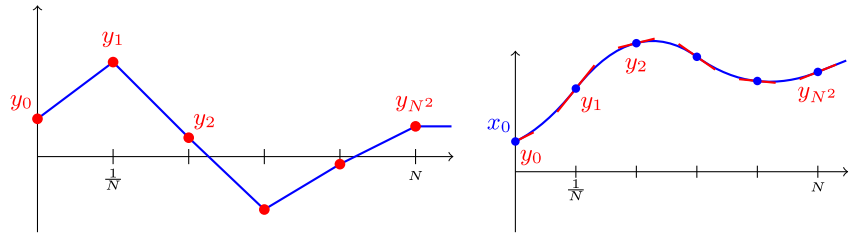

Figure 4. A continuous piecewise linear function v (left) and its integral, a C^1 piecewise quadratic function u (right). The data consist of an initial value x_0 and derivative values $y_0, \ldots, y_{N^2}$ at equispaced points.

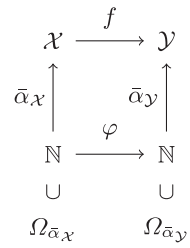

FIGURE 5. Tracking function.

DEFINITION 4.6
(**Tracking computability**).
Let $\mathcal{X}$ and $\mathcal{Y}$ be complete metric spaces with enumerated countable dense subsets $(\mathcal{X}_c, \alpha_{\mathcal{X}})$, $(\mathcal{Y}_c, \alpha_{\mathcal{Y}})$ and computability structures $(\Omega_{\bar{\alpha}_{\mathcal{X}}}, C_{\bar{\alpha}_{\mathcal{X}}}, \bar{\alpha}_{\mathcal{X}})$, $(\Omega_{\bar{\alpha}_{\mathcal{Y}}}, C_{\bar{\alpha}_{\mathcal{Y}}}, \bar{\alpha}_{\mathcal{Y}})$. Let $f : \mathcal{X} \rightharpoonup \mathcal{Y}$, and consider a function $\varphi : \mathbb{N} \rightharpoonup \mathbb{N}$.

We say that φ is a *tracking function* with respect to $(\alpha_{\mathcal{X}}, \alpha_{\mathcal{Y}})$, or an $(\alpha_{\mathcal{X}}, \alpha_{\mathcal{Y}})$-*tracking function*, for f, if

$$\begin{aligned} &\text{for all } c \in \Omega_{\bar{\alpha}_{\mathcal{X}}} \text{ with } \bar{\alpha}_{\mathcal{X}}(c) \in \operatorname{dom} f, \text{ we have that} \\ &c \in \operatorname{dom} \varphi \text{ and } \varphi(c) \in \Omega_{\bar{\alpha}_{\mathcal{Y}}} \text{ and } \bar{\alpha}_{\mathcal{Y}}(\varphi(c)) = f(\bar{\alpha}_{\mathcal{X}}(c)). \end{aligned} \tag{5}$$

When a function f has a recursive tracking function φ, we say that f is *tracking computable* with respect to $(\alpha_{\mathcal{X}}, \alpha_{\mathcal{Y}})$, or $(\alpha_{\mathcal{X}}, \alpha_{\mathcal{Y}})$-*computable*.

REMARK 4.7
(**Strict tracking computability**).
In the theory of tracking computability, stronger notions of *strict* tracking function and *strict* tracking computability are often also considered (see, e.g. [22, Definition 7.1.2]). A *strict tracking function*

is a tracking function ϕ such that, in addition to (5),

$$\begin{aligned} &\text{for all } c \in \Omega_{\tilde{\alpha}_{\mathcal{X}}} \text{ with } \bar{\alpha}_{\mathcal{X}}(c) \notin \operatorname{dom} f, \text{ we have that} \\ &c \notin \operatorname{dom} \varphi \text{ or } \varphi(c) \notin \Omega_{\tilde{\alpha}_{\mathcal{Y}}}. \end{aligned} \tag{6}$$

For total functions f, the concepts of tracking function and strict tracking function coincide, and hence, in particular, our proof that adders, multipliers, and integrators define tracking computable functions (Lemma 5.2) holds for strict tracking computability as well. However, for the case of continuous limits we only prove that it defines a tracking computable function. It would be interested to investigate whether we can define a strict tracking function for continuous limits, and more generally, whether, or under what conditions, the results in this paper extend to the stronger notion of strict tracking computability.

5 Computability of the input–output operator

The goal of this section is to demonstrate that the input-output operator of an LGPAC is tracking computable. We first show that this follows directly from the tracking computability of the basic modules.

LEMMA 5.1
(**Tracking computability of the input–output operator**).
Let $\mathcal{G}$ be a multityped GPAC with input–output operator $\Phi : \mathcal{I} \times \mathcal{M} \rightharpoonup \mathcal{M} \times \mathcal{O}$. Suppose that each of the modules occurring in $\mathcal{G}$ defines a tracking computable function. Then Φ is tracking computable.

PROOF. Let $M_1, \ldots, M_\ell$ be the modules appearing in $\mathcal{G}$, each defining a corresponding function $F_1, \ldots, F_\ell$, and having a corresponding tracking function $\varphi_1, \ldots, \varphi_\ell$. Note that Φ can be obtained from $F_1, \ldots, F_\ell$ via composition, projection and pairing. More formally, we can write $\Phi(x_1, \ldots, x_k) = (y_1, \ldots, y_\ell)$ where each $y_j = F_j(\mathbf{x}_j)$, and $\mathbf{x}_j$ is a subsequence of the inputs $x_1, \ldots, x_k$. Since composition, projection and pairing preserve tracking computability,[7] we obtain a recursive tracking function φ for Φ from $\varphi_1, \ldots, \varphi_\ell$. Thus, Φ is tracking computable. □

Hence, we only need to prove that each of the basic modules considered in Section 3 is tracking computable, which can be done under suitable assumptions.

LEMMA 5.2
(**Tracking computability of the LGPAC modules**).
Let $\mathcal{X}$ be a separable Fréchet space. Suppose that addition, scalar multiplication and pseudonorm evaluation are all tracking computable on $\mathcal{X}$. Let $\mathcal{Z} = C^1(\mathbb{T}, \mathcal{X})$ be the space of $\mathcal{X}$-streams with the computable structure induced by $\alpha_{\mathcal{X}}$, as in Example 4.4. Then

1. for each computable element $x \in \mathcal{X}$, the constant stream $u(t) = x$ is a computable element in $C^1(\mathbb{T}, \mathcal{X})$;
2. each of the nonconstant modules from Section 3 (adder, multiplier, integrator and continuous limit) defines a tracking computable function.

[7] The computability of basic algebraic operations is usually one of the first results to be proved for a model of computation. For example, in the framework of computable analysis, this is proved in [14, Section 0.4]; and in the framework of type-2 theory of effectivity, this is proved in [25, Section 2.1]. The techniques carry over to the tracking computability framework in this paper.

PROOF. We sketch the proof outline; additional technical details are given in the Appendix. Recall that an element $u \in \mathcal{Z}_c$ is described by a data tuple $(x_0, y_0, \ldots, y_{N^2})$ and can be encoded by $e = \langle N, \langle m_0, m'_0, \ldots, m'_{N^2}\rangle\rangle$, where $m_0, m'_0, \ldots, m'_{N^2}$ encode elements in $\mathcal{X}_c$.

Constants: Given an element $x \in \mathcal{X}_c$, the constant stream $u(t) = x$ is in $\mathcal{Z}_c$; in particular, it is encoded by $N = 1$ and the tuple $(x, 0, 0)$. Recall that, by assumption, we have $\alpha_{\mathcal{X}}(0) = 0$. Thus, given a code $c = \langle T, M\rangle$ for a computable $x \in \mathcal{X}$, we can consider the code $c' = \langle T', M\rangle$ in which $\{T'\}(j) = \langle 1, \langle\{T\}(j), 0, 0\rangle\rangle$. To verify that the same modulus of convergence works, let $x_j = \alpha_{\mathcal{X}}\{T\}(j)$ and $u_j = \alpha_{\mathcal{Z}}\{T'\}(j)$. Then, $u_j(t) \equiv x_j$, so that $u'_j(t) \equiv 0$ and

$$\|u_i - u_j\|_n = \|u_i(0) - u_j(0)\|_n + \sup_{0\le t\le n} \|u'_i(t) - u'_j(t)\|_n = \|x_i - x_j\|_n.$$

Hence, c' is a code for the desired constant stream, so that u is a computable element in $\mathcal{Z}$.

Addition: Essentially, we need to approximately compute addition at 'two levels'. At the 'first level', we create a procedure that receives codes e_1 and e_2 for computable $\mathcal{X}$-streams $u_1 = \alpha(e_1)$ and $u_2 = \alpha(e_2)$, as well as a natural number ℓ; it produces a code e_+ of some element $u_+ = \alpha(e_+)$ that approximates $u_1 + u_2$ to precision $2^{-\ell}$. This is done by building a large *common refinement*, i.e. codes for approximations $\tilde{u}_1, \tilde{u}_2$ of u_1, u_2 on a finer common grid. Each value in the new discretization can be seen as a convex combination of two consecutive values in the old discretization. Since addition and scalar multiplication are tracking computable in $\mathcal{X}$, these convex combinations can be approximated to arbitrarily high precision. Then, in order to compute the addition on the common refinement, we can simply compute the pointwise addition with sufficiently high precision.

At the 'second level', assume we have codes $c_1 = \langle T_1, M_1\rangle, c_2 = \langle T_2, M_2\rangle$ for computable elements u and v, respectively; we wish to find a code $c_+ = \langle T_+, M_+\rangle$ for their sum $w = u + v$. If we write $u_i = \alpha\{T_1\}(i)$, $v_i = \alpha\{T_2\}(i)$, $w_i = \alpha\{T_+\}(i)$, then the main idea is to define w_j as a (sufficiently good) approximation of $u_{k_1(j)} + v_{k_2(j)}$, for some choice of k_1 and k_2 (depending on the moduli of convergence $\{M_1\}$ and $\{M_2\}$) that ensures w_j is id-convergent to $u + v$.

Scalar multiplication: Compared to addition, there are two additional sources of error that we have to control. At the 'first level', we recall the product rule for derivatives, $(ru)'(t) = r(t)u'(t) + r'(t)u(t)$. After finding approximate $\tilde{r}$, $\tilde{u}$ on a large common refinement, we can approximately evaluate the above expression at equispaced points, as long as we are able to compute $\tilde{r}(j/N)$ and $\tilde{u}(j/N)$. Since $\tilde{r}$ and $\tilde{u}$ are piecewise quadratic, these can be retrieved by integration using the trapezoid rule $\tilde{x}_{j+1} \approx \tilde{x}_j + \frac{1}{2N}(\tilde{y}_j + \tilde{y}_{j+1})$, and hence computed to arbitrary precision. Yet another source of error appears in the analysis, since any approximation of ru is piecewise quadratic whereas ru itself is piecewise quartic (as functions of t). This additional error can be controlled by first finding an upper bound K_ℓ on $\|r\|_\ell$ and $\|u\|_\ell$ and then choosing a suitable large discretization $\bar{N}$.

At the 'second level', assume we have codes $c_1 = \langle T_1, M_1\rangle, c_2 = \langle T_2, M_2\rangle$ for sequences $r_i = \alpha\{T_1\}(i)$, $u_i = \alpha\{T_2\}(i)$ converging to computable elements r and u, respectively; we wish to find a code $c_\times = \langle T_\times, M_\times\rangle$ for a sequence $v_i = \alpha\{T_\times\}(i)$ converging to their product $v = ru$. Again, we define v_j to be an approximation of the product $r_{k_1(j)}u_{k_2(j)}$ to sufficiently high precision, computed at the 'first level'. By choosing a suitable $k_1(j), k_2(j)$ we can ensure (v_j) is id-convergent to ru.

Integration: The case of integration is quite similar to multiplication: if $x \in \mathcal{X}$ and $r \in C^1(\mathbb{T}, \mathbb{R})$, $u \in C^1(\mathbb{T}, \mathcal{X})$ are represented by the tuples of data $(p^1_0, q^1_0, \ldots, q^1_{N^2})$ and $(x^2_0, y^2_0, \ldots, y^2_{N^2})$, respectively, then the integral $w(t) = x + \int_0^t u(s)\mathrm{d}r(s)$ is a function with $w(0) = x$ and $w'(t) = u(t)r'(t)$. Since the values of $u(t)$ at equispaced points can be approximated by the trapezoid rule, this again yields a natural way to approximately compute a data tuple representation for w.

Continuous limit: If $u_n \in C^1(\mathbb{T}, \mathcal{X})$ is an effective Cauchy sequence converging to a stream $u \in C^1(\mathbb{T}, \mathcal{X})$ which in turn has an id-convergent limit $x \in \mathcal{X}$, then x equals $\lim_{t\to\infty}\lim_{n\to\infty} u_n(t)$.

Thus, a candidate for an approximation of x is $u_{k_n}(t_n)$, where t_n and k_n are large enough integers. By effectivizing this line of thought, we produce a tracking function for the continuous limit module as well. □

6 Computability of LGPAC-generable functions

In this section we prove the main result of this paper. The goal is to find out under which conditions the function generated by an LGPAC is tracking computable. We recall that, in our terminology, an LGPAC induces an operator and fixed point problem

$$\Phi : \mathcal{I} \times \mathcal{M} \rightharpoonup \mathcal{M} \times \mathcal{O}, \quad \Phi(\mathbf{u}^I, \mathbf{u}^M) = (\mathbf{u}^M, \mathbf{u}^O); \tag{7}$$

for the LGPAC to generate a valid function, we require the fixed point problem to be *well-posed* on $\mathcal{I}$, that is, (7) has a unique, continuous, solution map $F : \mathbf{u}^I \mapsto (\mathbf{u}^M, \mathbf{u}^O)$.

Our goal is to find conditions on Φ that imply that F is tracking computable. The idea is to find F by solving an *approximate fixed point problem*

Given $\mathbf{u}^I$ and $\epsilon > 0$, find $(\mathbf{u}^M, \mathbf{u}^O)$ such that $d(\Phi(\mathbf{u}^I, \mathbf{u}^M), (\mathbf{u}^M, \mathbf{u}^O)) < \epsilon$.

Moreover, from the point of view of tracking computability, we look for the desired $\mathbf{u}^M, \mathbf{u}^O$ in the enumerated, countable dense subset. Then, by using a sequence of ϵ converging to 0, and under an additional assumption on F (namely, we will require a notion of *effective well-posedness*; see Definition 6.2 below), this yields a sequence of $\mathbf{u}^M, \mathbf{u}^O$ converging to the desired $F(\mathbf{u}^I)$.

Let us now focus on the first step of this construction. Namely, we prove that it is possible to construct *approximate fixed points*.

LEMMA 6.1
Let $\mathcal{G}$ be an LGPAC with input-output operator $\Phi : \mathcal{I} \times \mathcal{M} \rightharpoonup \mathcal{M} \times \mathcal{O}$. Assume that Φ is tracking computable, and that $\mathcal{G}$ is well-posed on an open subset $U \subseteq \mathcal{I}$. Then there exists a computable procedure **FixPt** : $(n, \ell) \mapsto m$ such that, if n is the code for an element $\mathbf{u}^I = \bar{\alpha}_{\mathcal{I}}(n) \in U$ and $\ell \in \mathbb{N}$, then m is the code for an enumerated element $(\mathbf{u}^M, \mathbf{u}^O) = \alpha_{\mathcal{M}\times\mathcal{O}}(m) \in \mathcal{M} \times \mathcal{O}$; and also $d(\Phi(\mathbf{u}^I, \mathbf{u}^M), (\mathbf{u}^M, \mathbf{u}^O)) < 2^{-\ell}$.

PROOF. The procedure works as follows. For a given input n, ℓ, let us write $\mathbf{u}^I = \bar{\alpha}_{\mathcal{I}}(n)$. We perform the following dovetailing loop. First, guess an index $m \in \mathbb{N}$ for an element in $(\mathcal{M} \times \mathcal{O})_c$. Second, find m_1, m_2 such that $\alpha_{\mathcal{M}\times\mathcal{O}}(m) = (\alpha_{\mathcal{M}}(m_1), \alpha_{\mathcal{O}}(m_2))$, via the pairing bijections. For clarity, let us write $\mathbf{u}^M = \alpha_{\mathcal{M}}(m_1)$, $\mathbf{u}^O = \alpha_{\mathcal{O}}(m_2)$. Third, find n' such that $\bar{\alpha}_{\mathcal{I}\times\mathcal{M}}(n') = (\mathbf{u}^I, \mathbf{u}^M)$, using the code n for $\mathbf{u}^I$ and a code for the constant function $\{T_1\}(n) = m_1$. Fourth, find $m' = \varphi(n') = \langle T', M' \rangle$, where φ is a tracking function for Φ. Notice that

$$\bar{\alpha}_{\mathcal{M}\times\mathcal{O}}(m') = \bar{\alpha}_{\mathcal{M}\times\mathcal{O}}(\varphi(n')) = \Phi(\bar{\alpha}_{\mathcal{I}\times\mathcal{M}}(n')) = \Phi(\mathbf{u}^I, \mathbf{u}^M).$$

Fifth, find $m'' = \{T'\}(\{M'\}(\ell + 2))$. Since $\{M'\}$ is a module of convergence, it follows that for $k \geq \{M'\}(\ell + 2)$, one has

$$d(\alpha_{\mathcal{M}\times\mathcal{O}}(m''), \alpha_{\mathcal{M}\times\mathcal{O}}(\{T'\}(k))) < 2^{-\ell-2}.$$

In particular, since $\bar{\alpha}_{\mathcal{M}\times\mathcal{O}}(m')$ is the limit of $\alpha_{\mathcal{M}\times\mathcal{O}}(\{T'\}(n))$, then

$$d(\bar{\alpha}_{\mathcal{M}\times\mathcal{O}}(m'), \alpha_{\mathcal{M}\times\mathcal{O}}(m'')) \leq 2^{-\ell-2}.$$

For clarity, let us write $(\tilde{\mathbf{u}}^M, \tilde{\mathbf{u}}^O) = \alpha_{\mathcal{M}\times\mathcal{O}}(m'')$.

Finally, check if $d(\alpha_{\mathcal{M}\times\mathcal{O}}(m''), \alpha_{\mathcal{M}\times\mathcal{O}}(m)) < 2^{-\ell-1}$; if yes, then break the loop and return m. Observe that the distance function is tracking computable (to get a close enough approximation, it is enough to evaluate sufficiently but finitely many pseudonorms).

Observe that, for some values of m, the corresponding execution of the loop may not terminate. This may happen if $\bar{\alpha}_{\mathcal{I}\times\mathcal{M}}(n')$ is not an element in the domain of Φ, so that $\varphi(n')$ may be a divergent computation, or if the value of $d(\alpha_{\mathcal{M}\times\mathcal{O}}(m''), \alpha_{\mathcal{M}\times\mathcal{O}}(m))$ is exactly $2^{-\ell-1}$ (equality may not be a computable predicate). However, if a certain value of m happens to pass our test, then that value satisfies the desired property: indeed,

$$\begin{aligned} d(\Phi(\mathbf{u}^I, \mathbf{u}^M), (\mathbf{u}^M, \mathbf{u}^O)) &\leq d(\Phi(\mathbf{u}^I, \mathbf{u}^M), (\tilde{\mathbf{u}}^M, \tilde{\mathbf{u}}^O)) + d((\tilde{\mathbf{u}}^M, \tilde{\mathbf{u}}^O), (\mathbf{u}^M, \mathbf{u}^O)) \\ &= d(\bar{\alpha}_{\mathcal{M}\times\mathcal{O}}(m'), \alpha_{\mathcal{M}\times\mathcal{O}}(m'')) + d(\alpha_{\mathcal{M}\times\mathcal{O}}(m''), \alpha_{\mathcal{M}\times\mathcal{O}}(m)) \\ &< 2^{-\ell-2} + 2^{-\ell-1} = 2^{-\ell}. \end{aligned}$$

Moreover, such a value of m can always be found by our algorithm, due to our assumption that $\mathcal{G}$ is well-posed on U. To see this, let $\mathbf{u}^I = \bar{\alpha}_{\mathcal{I}}(n) \in U$. By well-posedness, there exists (a unique) $(\mathbf{u}^M_*, \mathbf{u}^O_*) \in \mathcal{M} \times \mathcal{O}$ with $\Phi(\mathbf{u}^I, \mathbf{u}^M_*) = (\mathbf{u}^M_*, \mathbf{u}^O_*)$, and thus $d(\Phi(\mathbf{u}^I, \mathbf{u}^M_*), (\mathbf{u}^M_*, \mathbf{u}^O_*)) = 0$. Now the left hand side of this equality is a continuous expression in $\mathbf{u}^M_*, \mathbf{u}^O_*$ (the continuity of Φ follows from the continuity of the module functions, and every metric d is continuous over its topology); thus there exists $\delta > 0$ such that for any $\mathbf{u}^M, \mathbf{u}^O \in \mathcal{M} \times \mathcal{O}$ one has

$$\text{if } d((\mathbf{u}^M_*, \mathbf{u}^O_*), (\mathbf{u}^M, \mathbf{u}^O)) < \delta \text{ then } d(\Phi(\mathbf{u}^I, \mathbf{u}^M), (\mathbf{u}^M, \mathbf{u}^O)) < 2^{-\ell-2}.$$

By density of the enumerated subset, there exists $m \in \mathbb{N}$ such that $\alpha_{\mathcal{M}\times\mathcal{O}}(m) = (\mathbf{u}^M, \mathbf{u}^O)$ with $d((\mathbf{u}^M_*, \mathbf{u}^O_*), (\mathbf{u}^M, \mathbf{u}^O)) < \delta$, and thus $d(\Phi(\mathbf{u}^I, \mathbf{u}^M), (\mathbf{u}^M, \mathbf{u}^O)) < 2^{-\ell-2}$, or in other words,

$$d(\bar{\alpha}_{\mathcal{M}\times\mathcal{O}}(m'), \alpha_{\mathcal{M}\times\mathcal{O}}(m)) < 2^{-\ell-2}.$$

Moreover, the value of m'', computed on step 4, will be such that

$$d(\bar{\alpha}_{\mathcal{M}\times\mathcal{O}}(m'), \alpha_{\mathcal{M}\times\mathcal{O}}(m'')) \leq 2^{-\ell-2},$$

and a simple application of the triangle inequality yields that

$$d(\alpha_{\mathcal{M}\times\mathcal{O}}(m''), \alpha_{\mathcal{M}\times\mathcal{O}}(m)) < 2^{-\ell-1},$$

so the condition on step 5 is met. Thus, the dovetailing loop will effectively succeed in finding a valid m. □

We have shown that it is possible to find approximate fixed points of the input-output operator. In the next step, we would like to argue that *approximate fixed points* are in fact 'near' *exact fixed points*, which is by no means a trivial statement (rather, there is extensive research on this problem; see, e.g. [8, 9]). Intuitively, we want to establish conditions on the input-output operator Φ (and the corresponding solution functional F) that effectively ensure the following: for each ϵ there is δ such that if $d(\Phi(\mathbf{u}^I, \mathbf{u}^M), (\mathbf{u}^M, \mathbf{u}^O)) < \delta$, then $d(F(\mathbf{u}^I), (\mathbf{u}^M, \mathbf{u}^O)) < \epsilon$ (see Fig 6 for an intuition). This is captured in the following notion, that we use as an assumption towards proving our main theorem.

DEFINITION 6.2
(**Effective well-posedness**).
Let $\mathcal{G}$, Φ, U be as in Definition 3.5, with $\mathcal{G}$ well-posed on U and generating some function F. We say that $\mathcal{G}$ is *effectively well-posed* on U if there is a computable modulus of convergence $M : \mathbb{N} \to \mathbb{N}$

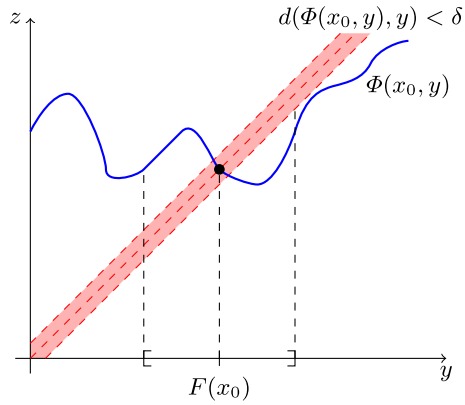

FIGURE 6. Approximate fixed points vs approximations of the fixed point. Intuitively, assume that the fixed point equation $\Phi(x, y) = y$ has a continuous solution operator $y = F(x)$. Then, in a neighborhood of x, approximate fixed points are 'near' the exact fixed point.

such that for all $\nu \in \mathbb{N}$ and all $\mathbf{u}^I \in U$, $\mathbf{u}^M \in \mathcal{M}$, $\mathbf{u}^O \in \mathcal{O}$,

$$d(\Phi(\mathbf{u}^I, \mathbf{u}^M), (\mathbf{u}^M, \mathbf{u}^O)) < 2^{-M(\nu)} \Rightarrow d(F(\mathbf{u}^I), (\mathbf{u}^M, \mathbf{u}^O)) < 2^{-\nu}. \tag{8}$$

REMARK 6.3
(**Well-posedness and effective well-posedness**).
The well-posedness of $\mathcal{G}$ implies that, for any $\mathbf{u}^I \in U$, $\mathbf{u}^M \in \mathcal{M}$, $\mathbf{u}^O \in \mathcal{O}$,

$$d(\Phi(\mathbf{u}^I, \mathbf{u}^M), (\mathbf{u}^M, \mathbf{u}^O)) = 0 \quad \text{iff} \quad d(F(\mathbf{u}^I), (\mathbf{u}^M, \mathbf{u}^O)) = 0.$$

Thus, effective well-posedness can be understood as an *effective strengthening* of this equivalence.

THEOREM 6.4
(**Tracking computability of LGPAC generable functions**).
Let $\mathcal{G}$ be an effectively well-posed LGPAC generating some function F on domain U. Suppose also that each of the modules in $\mathcal{G}$ are tracking computable. Then F is tracking computable.

PROOF. By Lemma 5.1, the input–output operator $\Phi : \mathcal{I} \times \mathcal{M} \rightharpoonup \mathcal{M} \times \mathcal{O}$ of $\mathcal{G}$ is tracking computable. By Lemma 6.1, there exists a procedure **FixPt** : $(e_I, \ell) \mapsto e_{\mathcal{M}\times\mathcal{O}}$ that maps codes of computable elements $\mathbf{u}^I \in U$ to $2^{-\ell}$-approximate fixed points. Let M_W be a computable modulus of convergence witnessing the effective well-posedness of $\mathcal{G}$. Then, given a code $c = \langle T, M\rangle$ of an element $\mathbf{u}^I \in U$, we can construct a code $\varphi(c) = \langle T', M'\rangle$ for $F(\mathbf{u}^I)$ by letting T' be a code for the function $\{T'\}(j) = \mathbf{FixPt}(c, M_W(j+1))$ and M' be a code for the identity function. Indeed, the above procedure is effective and, letting $(\mathbf{u}_j^M, \mathbf{u}_j^O) = \alpha\{T'\}(j)$, we have by construction that $d(\Phi(\mathbf{u}^I, \mathbf{u}_j^M), (\mathbf{u}_j^M, \mathbf{u}_j^O)) < 2^{-M_W(j+1)}$, so that $d(F(\mathbf{u}^I), (\mathbf{u}_j^M, \mathbf{u}_j^O)) < 2^{-j-1}$. In particular, $(\mathbf{u}_j^M, \mathbf{u}_j^O)$ is an id-convergent Cauchy sequence that converges to $F(\mathbf{u}^I)$. □

7 Some applications of Theorem 6.4

We proceed to give two applications of our main result.

1. Computability over continuous real functions. Let us consider the data space $\mathcal{X} = C(\mathbb{R})$ of continuous real functions. This can be considered a basic example of how the Shannon GPAC can be generalized beyond real-valued computation. Moreover, $C(\mathbb{R})$ is equipped with a multiplication operation $(fg)(x) = f(x)g(x)$, which naturally induces multiplication and integration over $C(\mathbb{R})$-streams, $\times(u,v)(t) = u(t)v(t)$ and $\int(c,u,v) = c + \int u\mathrm{d}v$. As one would expect, all these operations are tracking computable.

LEMMA 7.1
Let $\mathcal{X} = C(\mathbb{R})$ be the class of continuous real functions with the computability structure defined in Example 4.3. Then addition, scalar multiplication, multiplication, and pseudonorm evaluation are tracking computable operations on $\mathcal{X}$. Moreover, multiplication and integration over $C^1(\mathbb{T},\mathcal{X})$, defined as $\times(u,v)(t) = u(t)v(t)$ and $\int(c,u,v) = c + \int u\mathrm{d}v$ are also tracking computable operations.

PROOF. For addition, multiplication and integration, we can adapt the proof from Lemma 5.2. Note that, due to our choice of computability structure on $\mathbb{R}$ (Example 4.2), addition and multiplications by rationals can be performed *exactly* (on their codes), so a lot of the error analysis disappears. Moreover, functions in $C(\mathbb{R})$ are approximated by piecewise linear functions instead of the more complicated piecewise quadratic functions that we used in $C^1(\mathbb{T},\mathcal{X})$. Thus, the proofs become much simpler and we omit the details.

Now consider pseudonorm evaluation. Given a function $f \in \mathcal{X}_c$ via its code $e_1 = \langle N, \langle m_{-N^2}, \ldots, m_0, \ldots, m_{N^2}\rangle\rangle$, and an integer n, note that $\|f\|_n$ can be computed exactly: it simply corresponds to the maximum of the rational numbers $\alpha_{\mathbb{R}}(m_j)$, where j ranges either: between $-N^2$ and N^2 (if $N \leq n$); or between $-Nn$ and Nn (if $N \geq n$).

Next, let $c = \langle T, M\rangle$ be a code for a function in $\mathcal{X}$ and n be an integer. Let $f_j = \alpha_{\mathcal{X}}\{T\}(j)$. Define a code $\langle T_n, M_n\rangle$ where $\{T_n\}(j)$ is a code for the value $\|f_j\|_n$ (which can be computed exactly) and $\{M_n\}(\nu) = \{M\}(\nu + n)$. Note that for $i,j \geq \{M_n\}(\nu)$ we have that $d_{\mathcal{X}}(f_i,f_j) < 2^{-(\nu+n)}$ and hence $\|f_i - f_j\|_n < 2^{-\nu}$ (by Proposition 2.3). By the triangular inequality we conclude that $|\|f_i\|_n - \|f_j\|_n| < 2^{-\nu}$ as desired. □

Combining Lemma 5.2 and 7.1, we conclude that each nonconstant module on a multityped GPAC over $C(\mathbb{R})$ is tracking computable. Together with theorem 6.4, we obtain:

COROLLARY 7.2
Let $\mathcal{G}$ be a multityped GPAC with channels over $\mathbb{R}$ and $\mathcal{X} = C(\mathbb{R})$, constructed with the following types of modules: constants (over $\mathbb{R}$ and $\mathcal{X}$), adders (over $\mathbb{R}$-streams and $\mathcal{X}$-streams), multipliers $\times(u,v)$ and integrators $\int(c,u,v)$, (where each of u,v is either an $\mathbb{R}$-stream or an $\mathcal{X}$-stream), and continuous limits (over $\mathbb{R}$-streams and $\mathcal{X}$-streams). Suppose that each of the constant modules appearing in $\mathcal{G}$ is tracking computable, and that $\mathcal{G}$ is effectively well-posed on U, generating a function F. Then F is tracking computable.

2. Contracting operators. We show that the condition of effective well-posedness (Definition 6.2) is automatically achieved for contracting operators, which form an important class in fixed point theory. Formally, an input-output operator Φ is *contracting* if there is a constant $\lambda \in [0,1)$ such that

$$d(\Phi(\mathbf{u}^I,\mathbf{u}^M),\Phi(\mathbf{u}^I,\tilde{\mathbf{u}}^M)) \leq \lambda d(\mathbf{u}^M,\tilde{\mathbf{u}}^M).$$

LEMMA 7.3
Let $\mathcal{G}$ be a multityped GPAC and assume that its input-output operator Φ is contracting. Then $\mathcal{G}$ is well-posed iff it is effectively well-posed.

PROOF. Clearly, effective well-posedness implies well-posedness. For the converse direction, assume $\mathcal{G}$ is well-posed and let F be the function generated by $\mathcal{G}$. For each $\mathbf{u}^I \in \mathcal{I}$ define $\hat{\Phi} : \mathcal{M} \times \mathcal{O} \to \mathcal{M} \times \mathcal{O}$ as $\hat{\Phi}(\mathbf{u}^M, \mathbf{u}^O) = \Phi(\mathbf{u}^I, \mathbf{u}^M)$. In this way, $\hat{\Phi}$ is a contracting self-map on $\mathcal{M} \times \mathcal{O}$ with the same constant λ. As a consequence of the Banach fixed point theorem [15,Th. V.18], for any $(\mathbf{u}^M, \mathbf{u}^O) \in \mathcal{M} \times \mathcal{O}$ we have that $d(F(\mathbf{u}^I), (\mathbf{u}^M, \mathbf{u}^O)) \leq \frac{1}{1-\lambda} d(\hat{\Phi}(\mathbf{u}^M, \mathbf{u}^O), (\mathbf{u}^M, \mathbf{u}^O))$. Let us take $M(\nu) = \nu + C$ as a modulus of convergence, where C is any natural such that $2^{-C} \leq 1 - \lambda$. Then for any $\mathbf{u}^I \in \mathcal{I}, \mathbf{u}^M \in \mathcal{M}, \mathbf{u}^O \in \mathcal{O}$ such that $d(\Phi(\mathbf{u}^I, \mathbf{u}^M), (\mathbf{u}^M, \mathbf{u}^O)) < 2^{-M(\nu)}$, we get

$$d(F(\mathbf{u}^I), (\mathbf{u}^M, \mathbf{u}^O)) \leq \frac{1}{1-\lambda} d(\hat{\Phi}(\mathbf{u}^M, \mathbf{u}^O), (\mathbf{u}^M, \mathbf{u}^O)) < \frac{2^{-\nu-C}}{1-\lambda} < 2^{-\nu}.$$

□

The following corollary is immediate from Theorem 6.4 and Lemma 7.3.

COROLLARY 7.4
Let $\mathcal{G}$ be a well-posed LGPAC generating some function F on domain U. Suppose that each of the modules in $\mathcal{G}$ are tracking computable, and that the input-output operator Φ is contracting. Then F is tracking computable.

8 Discussion

In this paper we presented partial results towards a comparison between the GPAC model of computation and tracking computability on separable Fréchet spaces $\mathcal{X}$. Two important questions are left for further research.

1. Effective well-posedness. Our main result hinges on this extra assumption, allowing us to use approximate fixed points to obtain approximations of the exact fixed point. The question of whether this condition can be relaxed remains an open problem. Our difficulty stems from the usage of arbitrary data spaces $\mathcal{X}$, which in particular can be infinite-dimensional. In the case of $\mathcal{X} = \mathbb{R}^k$, i.e. finite-dimensional spaces, standard results in analysis (e.g. the Picard-Lindelöf Theorem, [4]) allow us to consider iterative methods to obtain such fixed points. Related to this observation, we have argued that effective well-posedness comes 'for free' when the input-output operator is contracting. It would be interesting to extend this argument to a larger class of 'typical' operators appearing in Analysis.

2. Converse of Theorem 6.4. Investigating under which conditions tracking computable functions are LGPAC-generable remains a major open problem. The most likely approach to answer this question may be to first simulate the behavior of a Turing machine (or any other discrete model of computation) in an analog network. As relevant literature, papers [1, 3, 5] provide a way to embed states, transitions, and the discrete evolution of a Turing machine into real numbers, continuous real functions, and the continuous evolution of a dynamical system respectively. With some care, their techniques may be adaptable to our framework.

We hope that in tackling these problems new insights can be acquired about the power of analog networks, and in particular the GPAC, as a model for analog computation.

Acknowledgements

The research of Diogo Poças was supported by the Alexander von Humboldt Foundation with funds from the German Federal Ministry of Education and Research (BMBF). The research of Jeffery Zucker was supported by the Natural Sciences and Engineering Research Council of Canada.

References

[1] O. Bournez, M. L. Campagnolo, D. S. Graça and E. Hainry. Polynomial differential equations compute all real computable functions on computable compact intervals. *Journal of Complexity*, **23**, 317–335, 2007.

[2] V. Bush. The differential analyzer. A new machine for solving differential equations. *Journal of the Franklin Institute*, **212**, 447–488, 1931.

[3] M. L. Campagnolo, C. Moore and J. F. Costa. Iteration, inequalities, and differentiability in analog computers. *Journal of Complexity*, **16**, 642–660, 2000.

[4] E. A. Coddington and N. Levinson. *Theory of Ordinary Differential Equations*. In McGraw-Hill, 1955.

[5] D. Graça, M. Campagnolo and J. Buescu. Robust simulations of Turing machines with analytic maps and flows. In *New Computational Paradigms, CiE 2005*, volume 3526 of Lecture Notes in Computer Science, pp. 169–179. Springer, 2005.

[6] Douglas R. Hartree. *Calculating Instruments and Machines*. Cambridge University Press, 1950.

[7] N. D. James and J. I. Zucker. A class of contracting stream operators. *The Computer Journal*, **56**, 15–33, 2013.

[8] U. Kohlenbach and B. Lambov. Bounds on iterations of asymptotically quasi-nonexpansive mappings. *BRICS Report Series*, **10**, 2003.

[9] U. Kohlenbach and L. Leuştean. Asymptotically nonexpansive mappings in uniformly convex hyperbolic spaces. *Journal of the European Mathematical Society*, **12**, 71–92, 2010.

[10] A. I. Mal'cev. Constructive algebras I. In *The metamathematics of algebraic systems: collected papers, 1936-1967*, pp. 148–212, North-Holland, 1971.

[11] D. Poças and J. Zucker. Approximability in the GPAC. *Logical Methods in Computer Science*, **15**, 2019.

[12] Diogo Poças. *Analog Computability with Differential Equations*. PhD Thesis, McMaster University, 2017.

[13] D. Poças and J. Zucker. Analog networks on function data streams. *Computability*, **7**, 301–322, 2018.

[14] M. Pour-El and I. Richards. *Computability in Analysis and Physics*. Springer, 1989.

[15] M. Reed and B. Simon. *Methods of Modern Mathematical Physics: Functional Analysis*. Academic Press, Inc, 1980.

[16] W. Rudin. *Principles of Mathematical Analysis*. In International Series in Pure and Applied Mathematics, 3rd edn. McGraw-Hill, 1976.

[17] C. Shannon. Mathematical theory of the differential analyser. *Journal Mathematical Physics*, **20**, 337–354, 1941.

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

[19] V. Stoltenberg-Hansen and J. V. Tucker. Effective algebras. In *Handbook of Logic in Computer Science*, vol. **4**, pp. 357–526. Oxford University Press, Oxford, 1995.

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

[21] William Thomson and Peter Tate. *Treatise on Natural Philosophy*, chapter Appendix B, pp. 479–508. Cambridge University Press, 2nd edn, 1880.

[22] J. V. Tucker and J. I. Zucker. Abstract versus concrete computation on metric partial algebras. *ACM Transactions on Computational Logic*, **5**, 611–668, 2004.
[23] J. V. Tucker and J. I. Zucker. Computable total functions, algebraic specifications and dynamical systems. *Journal of Algebraic and Logic Programming*, **62**, 71–108, 2005.
[24] J. V. Tucker and J. I. Zucker. Abstract versus concrete computability: The case of countable algebras. In *Logic Colloquium '03, Proceedings of the Annual European Summer Meeting of the Association for Symbolic Logic*, Helsinki, August 2003, volume 24 of Lecture Notes in Logic, pp. 377–408. Springer, 2006.
[25] K. Weihrauch. Computable Analysis: An Introduction. In *Texts in Theoretical Computer Science*. Springer, Berlin Heidelberg, 2000.

Appendix: Technical details in the proof of Lemma 5.2

Addition: Let e_1, e_2, ℓ be natural numbers, where e_1 and e_2 encode computable $\mathcal{X}$-streams $u_1 = \alpha(e_1)$ and $u_2 = \alpha(e_2)$. We need to show how to effectively compute a code e_+ of some element $u_+ = \alpha(e_+)$ that approximates $u_1 + u_2$ to precision $2^{-\ell}$, that is, such that $\|u_+ - (u_1 + u_2)\|_\ell < 2^{-\ell-2}$.

We know that u_1 and u_2 are given by some data tuples $(x_0^1, y_0^1, \ldots, y_{N_1^2}^1)$ and $(x_0^2, y_0^2, \ldots, y_{N_2^2}^2)$ respectively. First, we build a large common refinement, that is, a large discretization parameter $\bar{N}$ which is a multiple of both N_1 and N_2, and data tuples $(\tilde{x}_0^1, \tilde{y}_0^1, \ldots, \tilde{y}_{\bar{N}^2}^1)$, $(\tilde{x}_0^2, \tilde{y}_0^2, \ldots, \tilde{y}_{\bar{N}^2}^2)$ that correspond to approximations $\tilde{u}_1, \tilde{u}_2$ of u_1, u_2 on a finer grid. For example, if $\bar{N} = k \times N_1$, $\tilde{u}_1$ can be obtained by setting $\tilde{y}_{ki+\ell}^1 \approx \frac{k-\ell}{k} y_i^1 + \frac{\ell}{k} y_{i+1}^1$; each value in the new discretization is a convex combination of two consecutive values in the old discretization. Since addition and scalar multiplication are tracking computable in $\mathcal{X}$, these convex combinations can be approximated to arbitrarily high precision.

To compute the addition on the common refinement, we can simply add the pointwise values, that is, set $x_0^+ \approx \tilde{x}_0^1 + \tilde{x}_0^2$ and $y_j^+ \approx \tilde{y}_j^1 + \tilde{y}_j^2$. By computing these sums with sufficiently high precision, we have indeed produced the desired code e_+.

By the previous discussion, we have a procedure **add** : $(e_1, e_2, \ell) \mapsto e_+$ such that, for $u_1 = \alpha(e_1), u_2 = \alpha(e_2), u_+ = \alpha(e_+)$, we have $\|u_+ - (u_1 + u_2)\|_\ell < 2^{-\ell-2}$. Next, assume we have codes $c_1 = \langle T_1, M_1 \rangle, c_2 = \langle T_2, M_2 \rangle$ for computable elements u and v respectively; we wish to find a code $c_+ = \langle T_+, M_+ \rangle$ for their sum $w = u + v$. Let us introduce the notation $u_i = \alpha\{T_1\}(i)$, $v_i = \alpha\{T_2\}(i)$, $w_i = \alpha\{T_+\}(i)$.

We shall set $\{T_+\}(j) = \mathbf{add}(\{T_1\}(k_1(j)), \{T_2\}(k_2(j)), j)$, where $k_1(j) = \{M_1\}(2j+2)$ and $k_2(j) = \{M_2\}(2j+2)$. Intuitively, w_j is a (sufficiently good) approximation of $u_{k_1(j)} + v_{k_2(j)}$. Furthermore, we set M_+ as a code for the identity function. To show that (w_j) is id-convergent, fix ν and suppose that $i, j \geq \nu$. Observe that

$$\begin{aligned}\|w_i - w_j\|_\nu \leq & \|w_i - (u_{k_1(i)} + v_{k_2(i)})\|_\nu + \|u_{k_1(i)} - u_{k_1(j)}\|_\nu \\ & + \|v_{k_2(i)} - v_{k_2(j)}\|_\nu + \|w_j - (u_{k_1(j)} + v_{k_2(j)})\|_\nu.\end{aligned}$$

To bound the first term above, we observe that $\|w_i - (u_{k_1(i)} + v_{k_2(i)})\|_\nu \leq \|w_i - (u_{k_1(i)} + v_{k_2(i)})\|_i < 2^{-i-2} \leq 2^{-\nu-2}$; a similar argument holds for the fourth term. For the second term, note that by our choice of $k_1(\nu)$ we have $d(u_{k_1(i)}, u_{k_1(j)}) < 2^{-2\nu-2}$, and by Proposition 2.3 this implies $\|u_{k_1(i)} - u_{k_1(j)}\|_\nu < 2^{-\nu-2}$; similarly for the third term. Putting all this together yields $\|w_i - w_j\|_\nu < 2^{-\nu}$, which again by Proposition 2.3 implies $d(w_i, w_j) < 2^{-\nu}$, as

desired. A similar reasoning also proves that w_i converges to $u + v$. Hence, addition is tracking computable.

Scalar multiplication: In the same way as for addition, we show how to approximately compute the scalar multiplication at 'two levels'. At the 'first level', let e_1, e_2 be natural numbers encoding a computable $\mathbb{R}$-stream $r = \alpha(e_1)$ and $\mathcal{X}$-stream $u = \alpha(e_2)$, respectively. We need to show that we can compute the scalar multiplication ru to an arbitrary precision. In particular, we will show how to effectively compute, given a natural number ℓ, a code $e_\times$ of some element $u_\times = \alpha(e_\times)$ such that $\|u_\times - ru\|_\ell < 2^{-\ell-2}$.

We know that r and u are given by some data tuples $(p_0, q_0, \ldots, q_{N_1^2})$ and $(x_0, y_0, \ldots, y_{N_2^2})$ respectively. First, we effectively find an upper bound K_ℓ on the pseudonorms $\|r\|_\ell$ and $\|u\|_\ell$ by (approximately) computing the maximum of $|p_0|, |q_j|, \|x_0\|_\ell, \|y_0\|_\ell$ (by assumption, pseudonorm evaluation is tracking computable on $\mathcal{X}$).

Next, we construct a large common refinement, say $(\tilde{p}_0, \tilde{q}_0, \ldots, \tilde{q}_{\tilde{N}^2})$ and $(\tilde{x}_0, \tilde{y}_0, \ldots, \tilde{y}_{\tilde{N}^2})$, corresponding to approximations $\tilde{r}, \tilde{u}$ of r, u on a finer grid, as we did for addition. To compute the multiplication on the common refinement, we recall the product rule for derivatives, $(ru)'(t) = r(t)u'(t) + r'(t)u(t)$. To compute this expression at equispaced values of t, we must first find the values of $\tilde{r}(j/N), \tilde{u}(j/N)$. Since $\tilde{r}, \tilde{u}$ are piecewise quadratic, these can be recursively obtained by integration using the trapezoid rule,

$$\tilde{p}_{j+1} \approx \tilde{p}_j + \frac{1}{2N}(\tilde{q}_j + \tilde{q}_{j+1}), \qquad \tilde{x}_{j+1} \approx \tilde{x}_j + \frac{1}{2N}(\tilde{y}_j + \tilde{y}_{j+1}). \tag{A.1}$$

Again, $\tilde{p}_j, \tilde{x}_j$ can be approximated to arbitrarily high precision. Therefore, $\tilde{r}\tilde{u}$ can be approximated by the function $u_\times$ given by $(x_0^\times, y_0^\times, \ldots, y_{\tilde{N}^2}^\times)$, where $x_0^\times$ is (the approximating computation of) $\tilde{p}_0\tilde{x}_0$; and each $y_j^\times$ is (the approximating computation of) $\tilde{p}_j\tilde{y}_j + \tilde{q}_j\tilde{x}_j$.

There is one more error term appearing in our analysis, since $u_\times$ is piecewise quadratic whereas ru is piecewise quartic (as functions of t). To describe an effective bound on the approximation error $\|u_\times - ru\|_\ell$, we need to take into account: the approximation errors for the refinement and the multiplications over $\mathcal{X}_c$, the upper bound K_ℓ on $\|r\|_\ell$ and $\|u\|_\ell$; the consecutive differences $\max \|\tilde{q}_{j+1} - \tilde{q}_j\|_n, \max \|\tilde{y}_{j+1} - \tilde{y}_j\|_n$; and the discretization $\tilde{N}$. Ultimately, we can bound this error in an effective way by choosing $\tilde{N}$ large enough.

By the previous discussion, we have a procedure **mult** : $(e_1, e_2, \ell) \mapsto e_\times$ such that, for $r = \alpha(e_1), u = \alpha(e_2), u_\times = \alpha(e_\times)$, we have $\|u_\times - ru\|_\ell < 2^{-\ell-2}$. At the 'second level', assume we have codes $c_1 = \langle T_1, M_1\rangle, c_2 = \langle T_2, M_2\rangle$ for computable elements r and u respectively; we wish to find a code $c_\times = \langle T_\times, M_\times\rangle$ for their product $v = ru$. Let us introduce the notation $r_i = \alpha\{T_1\}(i)$, $u_i = \alpha\{T_2\}(i)$, $v_i = \alpha\{T_\times\}(i)$.

First, for any $\nu \in \mathbb{N}$, we can effectively find a *uniform bound* $K(\nu)$ such that $\|r_i\|_\nu, \|u_i\|_\nu < K(\nu)$ independently of i. This is because, letting $\mu = \{M_1\}(\nu)$, we know that for $i > \mu$ one has $d(r_i, r_\mu) < 2^{-\nu}$ and hence $\|r_i - r_\mu\|_\nu < 1$ by Proposition 2.3, so that $\|r_i\|_\nu < \|r_\mu\|_\nu + 1$. On the other hand, we can approximately compute $\|r_i\|_\nu$ for each of the finitely many $i \le \mu$. A similar analysis holds for $\|u_i\|_\nu$. Taking (a sufficiently close approximation of) the maximum of these values gives the desired uniform bound.

Next, observe that for any $r, \tilde{r} \in \mathbb{R}, x, \tilde{x} \in \mathcal{X}, \nu \in \mathbb{N}$, we have $\|rx - \tilde{r}\tilde{x}\|_\nu \le |r|\|x - \tilde{x}\|_\nu + |r - \tilde{r}|\|\tilde{x}\|_n$; together with (3), we can derive the useful bound

$$\|r_{i_1}u_{j_1} - r_{i_2}u_{j_2}\|_\nu \le (\nu + 1)K(\nu)\left(\|r_{i_1} - r_{i_2}\|_\nu + \|u_{j_1} - u_{j_2}\|_\nu\right). \tag{A.2}$$

We are now in condition to describe how to compute $\{T_\times\}(\nu)$ for a given ν. First, find a uniform bound $K(\nu)$ as described above. Second, find an integer C such that $2^C > K(\nu)(\nu + 1)$. Third,

compute $k_1(\nu) = \{M_1\}(2\nu + C + 2)$ and $k_2(\nu) = \{M_2\}(2\nu + C + 2)$. Finally, return

$$\{T_\times\}(\nu) = \mathbf{mult}(\{T_1\}(k_1(\nu)), \{T_2\}(k_2(\nu)), \nu).$$

Intuitively, this means that v_i is a (sufficiently good) approximation of $r_{k_1(i)}u_{k_2(i)}$. We show that the sequence v_i constructed in this way is id-convergent. Fix ν and suppose that $i,j \geq \nu$. Observe that

$$\|v_i - v_j\|_\nu \leq \|v_i - r_{k_1(i)}u_{k_2(i)}\|_\nu + \|r_{k_1(i)}u_{k_2(i)} - r_{k_1(j)}u_{k_2(j)}\|_\nu + \|r_{k_1(j)}u_{k_2(j)} - v_j\|_\nu.$$

The first term above, by construction, can be bounded as $\|v_i - r_{k_1(i)}u_{k_2(i)}\|_\nu \leq \|v_i - r_{k_1(i)}u_{k_2(i)}\|_i < 2^{-i-2} \leq 2^{-\nu-2}$, and similarly for the third term. In order to bound the second term, note that by our choice of $k_1(\nu)$ we have that $d(r_{k_1(i)}, r_{k_1(j)}) < 2^{-2\nu-2-C}$. By Proposition 2.3, this implies $\|r_{k_1(i)} - r_{k_1(j)}\|_\nu < 2^{-\nu-2-C} < \frac{2^{-\nu-1}}{2K(\nu)(\nu+1)}$. A similar bound holds for $\|u_{k_2(i)} - u_{k_2(j)}\|_\nu$. Putting these in (2) yields $\|r_{k_1(i)}u_{k_2(i)} - r_{k_1(j)}u_{k_2(j)}\|_\nu < 2^{-\nu-1}$. Thus, we conclude that $\|v_i - v_j\|_\nu < 2^{-\nu-2} + 2^{-\nu-1} + 2^{-\nu-2} = 2^{-\nu}$, and hence $d(v_i, v_j) < 2^{-\nu}$, i.e. v_i is id-convergent. A similar reasoning proves that v_i converges to ru. Hence, the above describes a tracking function for scalar multiplication.

Continuous limit: Let $u \in \mathcal{Z}_c$ be represented by the tuple $(x_0, y_0, \ldots y_{N^2})$, where each $x_0, y_j \in \mathcal{X}_c$. We first observe that, for any natural number $n \in \mathbb{N}$, the value of $u(n)$ can be approximated as

$$u(n) \approx \begin{cases} x_{nN} & \text{if } n \leq N; \\ x_{N^2} + (N-n)y_{N^2} & \text{if } n \geq N, \end{cases}$$

where the x_j are again recursively obtained via the trapezoid rule. Consequently, one can devise a computable procedure $\mathbf{eval} : (e, n, \ell) \mapsto e_{\text{eval}}$ such that, given a code e of some element $u = \alpha_{\mathcal{Z}}(e)$ and natural numbers n, ℓ, it produces a code e_{eval} of some element $x = \alpha_{\mathcal{X}}(e_{\text{eval}})$ with $d(x, u(n)) < 2^{-\ell}$; i.e. x approximates $u(n)$ within an error of $2^{-\ell}$.

Now let $c = \langle T, M\rangle$ be a code for an effective Cauchy sequence $u_j = \alpha_{\mathcal{Z}}\{T\}(j)$ in $\mathcal{Z}_c$ converging to a computable element $u \in C^1(\mathbb{T}, \mathcal{X})$. We want to compute a code $c_\infty = \langle T_\infty, M_\infty\rangle$ for an effective Cauchy sequence $x_j = \alpha_{\mathcal{X}}\{T_\infty\}(j)$ in $\mathcal{X}_c$ converging to the limit $x = \mathcal{L}u = \lim_{t\to\infty} u(t) \in \mathcal{X}$.

The idea is to define $\{T_\infty\}(j) = \mathbf{eval}(\{T\}(k_j), t_j, \ell_j)$, for a suitable choice of $\ell_j = j + 3$, $t_j = j + 2$ and $k_j = \{M\}(3j + 5)$. To prove that (x_j) is an id-convergent Cauchy sequence, let $\nu \in \mathbb{N}$ be given, and suppose that $i,j \geq \nu$. By applying the triangular inequality, $d_{\mathcal{X}}(x_i, x_j)$ is upper bounded as

$$\begin{aligned} d_{\mathcal{X}}(x_i, x_j) \leq & d_{\mathcal{X}}(x_i, u_{k_i}(t_i)) + d_{\mathcal{X}}(u_{k_i}(t_i), u(t_i)) + d_{\mathcal{X}}(u(t_i), u(t_j)) \\ & + d_{\mathcal{X}}(u(t_j), u_{k_j}(t_j)) + d_{\mathcal{X}}(u_{k_j}(t_j), x_j). \end{aligned}$$

By our choice of $\ell_j = j + 3$ we immediately get that $d_{\mathcal{X}}(x_i, u_{k_i}(t_i)) < 2^{-\nu-3}$ and $d_{\mathcal{X}}(u_{k_j}(t_j), x_j) < 2^{-\nu-3}$. Since u is an id-convergent Cauchy stream, and by our choice of $t_j = j+2$, we can also bound $d_{\mathcal{X}}(u(t_i), u(t_j)) < 2^{-\nu-2}$. Next we need to handle the terms $d_{\mathcal{X}}(u_{k_i}(t_i), u(t_i))$ and $d_{\mathcal{X}}(u_{k_j}(t_j), u(t_j))$, which amounts to show that $k_j = \{M\}(3j + 5)$ is suitably large.

Indeed, observe that $d_{\mathcal{Z}}(u_{k_j}, u) \leq 2^{-3j-5} = 2^{-3t_j+1}$. Using Proposition 2.3 then yields $\|u_{k_j} - u\|_{t_j} \leq 2^{-2t_j+1}$, and using (3) we have[8]

$$\|u_{k_j}(t_j) - u(t_j)\|_{t_j} \leq \frac{t_j}{2^{t_j-1}} 2^{-t_j} \leq 2^{-t_j}.$$

Once more by Proposition 2.3 we get $d_{\mathcal{X}}(u_{k_j}(t_j), u(t_j)) \leq 2^{-t_j} \leq 2^{-\nu-2}$. The same reasoning also gives the bound $d_{\mathcal{X}}(u_{k_i}(t_i), u(t_i)) \leq 2^{-\nu-2}$. Combining all these bounds yields $d_{\mathcal{X}}(x_i, x_j) < 2^{-\nu}$, so

[8] Observe that $t_j \leq 2^{t_j-1}$ for any $t_j = j + 2 \geq 2$.

that (x_j) is an id-convergent Cauchy sequence. In particular, we can take M_∞ to be a code for the identity function.

This construction shows that $c = \langle T, M\rangle \mapsto c_\infty = \langle T_\infty, M_\infty\rangle$ is an effective procedure. We also proved that, for all $j \in \mathbb{N}$, $d_{\mathcal{X}}(x_j, u(t_j)) < 2^{-j-3} + 2^{-j-2}$, implying that $\lim_j x_j = \lim_t u(t)$; hence c_∞ encodes an effective Cauchy sequence converging to $\mathcal{L}u$ as desired.

Received 30 May 2020