

Computability of analog networks

J.V. Tucker

*Department of Computer Science,
University of Wales, Swansea SA2 8PP, Wales*
J.V.Tucker@swansea.ac.uk

J.I. Zucker*

*Department of Computing and Software,
McMaster University, Hamilton, Ont. L8S 4L7, Canada*
zucker@mcmaster.ca

(April 15, 2008)

Abstract

We define a general concept of a network of analog modules connected by channels, processing data from a metric space A , and operating with respect to a global continuous clock \mathbb{T} . The inputs and outputs of the network are continuous streams $u : \mathbb{T} \rightarrow A$, and the input-output behaviour of the network with system parameters from A is modelled by a function $\Phi : A^r \times \mathcal{C}[\mathbb{T}, A]^p \rightarrow \mathcal{C}[\mathbb{T}, A]^q$ ($p, q > 0, r \geq 0$), where $\mathcal{C}[\mathbb{T}, A]$ is the set of all continuous streams equipped with the compact-open topology. We give an equational specification of the network, and a semantics which involves solving a fixed point equation over $\mathcal{C}[\mathbb{T}, A]$ using a contraction principle based on the fact that $\mathcal{C}[\mathbb{T}, A]$ can be approximated locally by metric spaces. We show that if the module functions are continuous then so is the network function Φ . We analyse in detail two case studies involving mechanical systems. Finally, we introduce a custom-made concrete computation theory over $\mathcal{C}[\mathbb{T}, A]$ and show that if the module functions are concretely computable then so is Φ .

Key words and phrases: analog computation, analog computing, analog networks, continuous time, continuous streams

* The research of the second author was supported by a grant from the Natural Sciences and Engineering Research Council (Canada)

1 Introduction

Digital and analog computation and communication both process infinite data such as real numbers, wave forms, signals, timed streams of data, and various other kinds of functions. The data invariably originate as physical measurements from the world's work.

In digital computation and communication, data are ultimately discrete. Each datum is representable by finitely many symbols, and the set of data is countable. Data representations are often made from strings of bits $\{0, 1\}$, or of other finite alphabets, and computations are expressed by algorithms computing functions on these strings. In the case of nondeterministic algorithms, the functions are multivalued. Digital computation is *exact* in the following sense: *given exact finite data as input, an exact computation returns exact finite data as output*. The finite input may be an approximation to some question, in which case the finite output is an approximation to an answer. The quality of the digital computation is determined by the level of control of the errors in the approximations presented to and propagated by the algorithm. Indeed, the above notion of exactness can be lifted to digital computation on infinite data, *i.e.*, computations from finite approximations of the input to finite approximations of the output, by calling such a computation “exact” when the computed function from input approximations to output approximations is accurate to *any* error margin. Digital computation is fundamentally computation by *algorithms*, which operate on symbols in discrete time.

In analog computation and communication, data is continuous. A datum can require an infinite symbolic representation, and the set of all representations is uncountable. Often the data are made from real numbers, and the functions computed are of the form $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$; these may be partial and/or many-valued.

Analog computation, as conceived by Lord Kelvin [TT80], Vannevar Bush [Bus31], and Douglas Hartree [Har50], 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., and the analog devices that process these representations are made from mechanical or electro-mechanical or electronic components [Hol96, Joh96, Sma96]. Here experimental procedures applied to the machine, especially measurements, play a special role. The inexactness of the measurement means that only an approximate input can be measured and presented to the analog device, and only an approximate output can be measured and returned from it. (In practice, an error of within 1% was easily achieved.)

The exact theoretical values of the analog input and output are unknown — and, perhaps, unknowable. Analog computation is a form of *computation by experimental procedures*. In general, analog devices are based on physical technology that operates in continuous time.

Starting in the 1930s, classical computability theory has matured into a comprehensive and mathematically deep *theory of digital computation*. Since its creation, Turing computability and its equivalents have become the standard for what we mean by computation. The subject continues to develop in new directions and applications [Gri99]. Of particular relevance is Computable Analysis, where the theory is applied to computable functions on real numbers, Banach spaces, and, more generally, metric and topological spaces [PER89,

Wei00].

The *theory of analog computation* is less developed. A general purpose analog computer (GPAC) was introduced by Shannon [Sha41] as a model of Bush's Differential Analyzer [Puc96]. Shannon discovered that a function can be generated by a GPAC if, and only if, it is differentially algebraic, but his proof was incomplete. A basic study was made by Marian Pour-El [PE74] who gave some good characterisations of the analog computable functions, focusing on the classic analog systems built from adders, multipliers, integrators etc. This yielded a stronger model and a new proof of the Shannon's equivalence (and some new gaps, corrected by Lipshitz and Rubel [LR87]). Using this characterisation in terms of algebraic differential equations, Pour-El showed that these analog models do not compute all computable functions on the reals.

Recently, the theory of analog computing has been boosted by Christopher Moore with some very general mathematical models [Moo96]. These models, using schemes rather like Kleene's [Kle52], but with primitive recursion replaced by integration and others added, can define functions beyond the class of computable functions on the reals. Félix Costa and his colleagues and students [GC03] have presented an improved model extending GPAC, and shown this to be equivalent with a subclass of Moore's functions (those defined by composition and integration). They have also presented some fine results concerning analog complexity classes [CMC02, MC04].

Technological interest in analog computing continues, for example, with such work as that of Jonathan Mills at the University of Indiana.¹

The revival of interest in analog computing is motivated by the search for solutions to old unsolved problems, and for new models of computation based on physical theories and technologies. For instance, analog computation, broadly conceived, provides a basis for a general theory of *analog field computation*, in which the primary data objects are scalar fields.

We present two theoretical questions about analog technology:

1. **Technology specification:** What are the characteristics of data, processing units, transmissions, system architecture, system operation and measurement that make up a technology for analog computation?
2. **Technology classification:** Given a technology that builds systems from physical components, do these analog systems produce, by measurements, the same functions than those computed by algorithms?

Thanks to Shannon and Pour-El and the improvements of the later authors mentioned, we have an example of one possible precise formulation of, and negative answer to, these technology questions. Their models are based on the traditional technological components of analog computing up until the 1960s (adders, integrators, etc.). However, even for the case of traditional analog technologies, the conceptual basis is not sufficiently clear to answer even the first question fully. In this paper we will address these questions in some generality.

¹<http://www.cs.indiana.edu/~jwmills/ANALOG.NOTEBOOK/klm/klm.html>

We begin, in Section 2, by defining a general concept of an analog network. This is a network of analog processing units connected by channels, processing data from a metric space A , and operating in continuous time, measured by a global clock \mathbb{T} , which is modelled by the set of non-negative real numbers. The input and output of a network are continuous streams $u: \mathbb{T} \rightarrow A$.

Let $\mathcal{C}[\mathbb{T}, A]$ be the set of all continuous streams equipped with the compact-open topology. The input-output behaviour of a network with system parameters from A^r is modelled by a function of the form

$$\Phi: A^r \times \mathcal{C}[\mathbb{T}, A]^p \rightarrow \mathcal{C}[\mathbb{T}, A]^q$$

where $\mathcal{C}[\mathbb{T}, A]$ is the set of all continuous streams of data from A . An analog network is *designed* according to some physical theory, but it will be *used* to compute by means of measurements on the network. We propose that the units satisfy an important physically motivated condition: *causality*. We show how to give an equational specification of the network.

In Section 3, we also propose a *stability condition* on the behaviour of the network, partially motivated by experimental procedure. We give a semantics for an equational specification of the network satisfying the causality condition. This involves solving a fixed point equation over $\mathcal{C}[\mathbb{T}, A]$ using a custom-made *contraction principle*, based on the fact that $\mathcal{C}[\mathbb{T}, A]$ can be locally approximated by metric spaces. This is an extension of the well-known Banach fixed point theorem for metric spaces [Eng89]. We derive the continuity of Φ from the continuity of the module functions. Hence, we have a conceptual and mathematical model of what it means for a network to be well-posed, and, therefore, for a function to be *computable by measurements on an analog system*.

In Section 4 we analyse in detail two case studies of analog computations, using mechanical systems in which data are represented by displacement, velocity and acceleration. Our aim is give informative and complete case studies of our general model.

In Section 5 we compare analog and digital computation. For this we introduce a custom-made concrete (algorithmic) computation theory over $\mathcal{C}[\mathbb{T}, A]$. The theory is *concrete* in the sense that it is based on choosing particular representations of data [SHT99, TZ04, TZ05, TZ06]. This is, again, an extension to the non-metric space $\mathcal{C}[\mathbb{T}, A]$ of the theory of concrete computations on metric algebras [TZ04]. We prove the following “soundness theorem” for analog, relative to concrete, computation.

Theorem. *If the functions defined by the components of an analog network are all concretely computable, then so is the function defined by the whole network.*

In particular, the results for traditional analog systems (based on real numbers and integrators etc.) can be easily derived. Settling a converse to these theorems, *i.e.*, completeness of analog computation, would be of great interest. (For work related to completeness using other models of computation, see [GCB05, BCGH06, BCGH07].)

We have also studied computation on discrete time streams [TZ94], and networks that process discrete time in streams. In [TT91] we develop a theory of synchronous algorithms

(SCAs) that generalise standard algorithmic discrete time models of computer hardware (microprocessors, systolic algorithms) and spatially extended dynamical systems (cellular automata, coupled map lattices, and neural networks). Mathematically, the present work can be seen as a generalisation of SCA’s to continuous time.

We are grateful to an anonymous referee for some very helpful comments.

2 Analog networks

An *analog network* N consists of a number of *modules* and *channels* computing and communicating with data from a topological algebra A . We make this idea precise in several stages.

2.1 Data, time and streams

Assume we are working with data from a *complete metric space* (A, d_A) . The network operates in continuous time \mathbb{T} , modelled by the set of non-negative reals with its usual metric topology. The channels carry signals in the form of *continuous streams* of data from A , represented as continuous functions $u: \mathbb{T} \rightarrow A$. Typical examples of streams are signals given by voltages as functions of time.

Let $\mathcal{C}[\mathbb{T}, A]$ be the set of continuous streams on A , with the *compact-open topology* [Eng89, §3.4]. We discuss properties of this topology, and equivalent formulations, below (Section 3).

2.2 Modules

A *module* M has finitely many *input channels* $\alpha_1, \dots, \alpha_{k_M}$ ($k_M \geq 1$), one *output channel* β (Figure 1), and locations for some *parameters* (not shown).

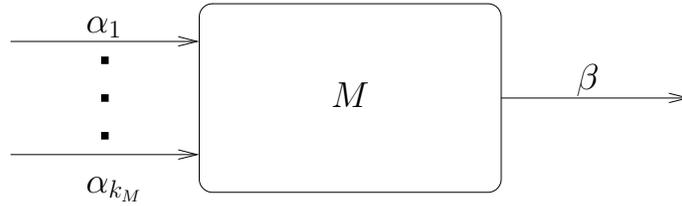


FIGURE 1: A module

Each module M is specified by a total function with $k_M > 0$ *input streams*, $l_M \geq 0$ *input parameters* and one *output stream*, which it “computes”:

$$F_M: \mathcal{C}[\mathbb{T}, A]^{k_M} \times A^{l_M} \rightarrow \mathcal{C}[\mathbb{T}, A].$$

For inputs $\mathbf{u} = (u_1, \dots, u_{k_M}) \in \mathcal{C}[\mathbb{T}, A]^{k_M}$ and parameters $\mathbf{c} = (c_1, \dots, c_{l_M}) \in A^{l_M}$, we write $F_M(\mathbf{u}, \mathbf{c}) = v$, where $v \in \mathcal{C}[\mathbb{T}, A]$ is the output

Examples 2.2.1. In classical analog computing ($A = \mathbb{R}$), typical module operations are:

- *Pointwise addition* of two streams:

$$F_M(u_1, u_2)(t) = u_1(t) + u_2(t).$$

This has 2 input streams, and no parameters.

- *Pointwise scalar multiplication* of a stream by a constant “scalar”:

$$F_M(u, c)(t) = c \cdot u(t).$$

This has 1 input stream u and 1 parameter c .

- *Integration*:

$$F_M(u, c)(t) = \left(\int_0^t u \right) + c.$$

This has 1 input stream and 1 parameter (the constant of integration), typically associated with the initial value $v(0)$ of the output $v(t)$.

- *Stieltjes integration*:

$$F_M(u_1, u_2, c)(t) = \left(\int_0^t u_1 du_2 \right) + c.$$

This has 2 input streams and 1 parameter (the constant of integration). Note that this is a partial operation, and treatment of it is therefore deferred to another paper. The stream u_2 must be *absolutely continuous* on $[0, t]$ for all $t > 0$, or equivalently, it must itself be a definite integral:

$$u_2(t) = \left(\int_0^t w \right) + u_2(0).$$

A *sufficient condition* for this is that u_2 have *bounded variation* on $[0, t]$ for all $t > 0$ [Roy63].

An important property of module functions is *causality*², which we now define.

Definition 2.2.3 (Causality of module functions). The output is “causally” related to the inputs, in the sense that the output at any time depends only on the inputs up to that time. Precisely:

Caus: For $\mathbf{u}, \mathbf{v} \in \mathcal{C}[\mathbb{T}, A]^{k_M}$, $\mathbf{c} \in A^{l_M}$ and $t \geq 0$:

$$\mathbf{u} \upharpoonright_{[0,t]} = \mathbf{v} \upharpoonright_{[0,t]} \implies F_M(\mathbf{u}, \mathbf{c})(t) = F_M(\mathbf{v}, \mathbf{c})(t).$$

Here we are using notation for vector restriction:

$$\mathbf{u} \upharpoonright_{[0,t]} =_{df} (u_1 \upharpoonright_{[0,t]}, \dots, u_k \upharpoonright_{[0,t]}).$$

We will also write $\mathbf{u} \upharpoonright_t$ for $\mathbf{u} \upharpoonright_{[0,t]}$ for $t \geq 0$.

²called *retrospectiveness* in [Tra66, Rab03]

Remarks 2.2.4 (Causality condition).

(a) By continuity of streams, **Caus** is equivalent to the apparently stronger condition:

$$\mathbf{u}_1 \upharpoonright_{[0,t)} = \mathbf{u}_2 \upharpoonright_{[0,t)} \implies F_M(\mathbf{u}_1, \mathbf{c})(t) = F_M(\mathbf{u}_2, \mathbf{c})(t).$$

- (b) In either version, the causality condition depends on an assumption of *instantaneous response* of the modules, and hence of the network. Contrast this with the unit time delay with discrete networks, *i.e.*, SCAs [HTT88, TT91].
- (c) All the common module operations, including the standard examples listed in 2.2.1, satisfy **Caus**.

In fact the functions we typically encounter as module functions (apart from integration) satisfy a stronger condition than causality, *i.e.*, *pointwise definability*, in the following sense.

Definition 2.2.5 (Pointwise definable functions). A function $F: \mathcal{C}[\mathbb{T}, A]^k \times A^l \rightarrow \mathcal{C}[\mathbb{T}, A]$ is *pointwise definable* if there is a function $f: A^k \times A^l \rightarrow A$ such that for all $\mathbf{u}, \mathbf{c}, t \geq 0$,

$$F(\mathbf{u}, \mathbf{c})(t) = f(\mathbf{u}(t), \mathbf{c}).$$

Lemma 2.2.6. *If F is pointwise definable by f , then it satisfies **Caus**.*

2.3 Network architecture

Consider now (Figure 2) a network N with m modules M_1, \dots, M_m and m channels $\alpha_1, \dots, \alpha_m$. Each module M_i ($i = 1, \dots, m$) has some *input channels* $\alpha_{i_1}, \dots, \alpha_{i_{k_i}}$ ($k_i > 0$), which are the outputs of modules $M_{i_1}, \dots, M_{i_{k_i}}$ respectively, some (*local*) *parameter locations* $c_{i_1}, \dots, c_{i_{l_i}}$ ($l_i \geq 0$) and one *output channel* α_i . The module M_i in the network N is specified by the function

$$F_i = F_{M_i}: \mathcal{C}[\mathbb{T}, A]^{k_i} \times A^{l_i} \rightarrow \mathcal{C}[\mathbb{T}, A].$$

The network N itself has p *input channels* and q *output channels* ($p, q \leq m$). We assume that the first p modules M_1, \dots, M_p are instances of the identity module $M_{\mathbf{I}}$. They function as *input ports*, and the p network input channels are actually their output channels $\alpha_1, \dots, \alpha_p$. (This is for the sake of uniformity of notation, to allow us to assume that each channel of N is the output channel of some module; *cf.* Figure 2). The remaining (non-trivial) modules of the network are M_{p+1}, \dots, M_m .

Thus for $i = 1, \dots, m$, the channel α_i is an output channel for module M_i . As stated above, the first p of these, $\alpha_1, \dots, \alpha_p$, are also the p network input channels. The q network output channels β_1, \dots, β_q are q of the m network channels, say $\beta_i = \alpha_{j_i}$ for $i = 1, \dots, q$.

There are also locations for the *global* or *network parameters* $\mathbf{c} = (c_1, \dots, c_r)$ ($r \geq 0$), which include all the local parameters of all the modules in N . Each module M_i selects its own list of local parameters $(c_{i_1}, \dots, c_{i_{l_i}})$ from the global list \mathbf{c} .

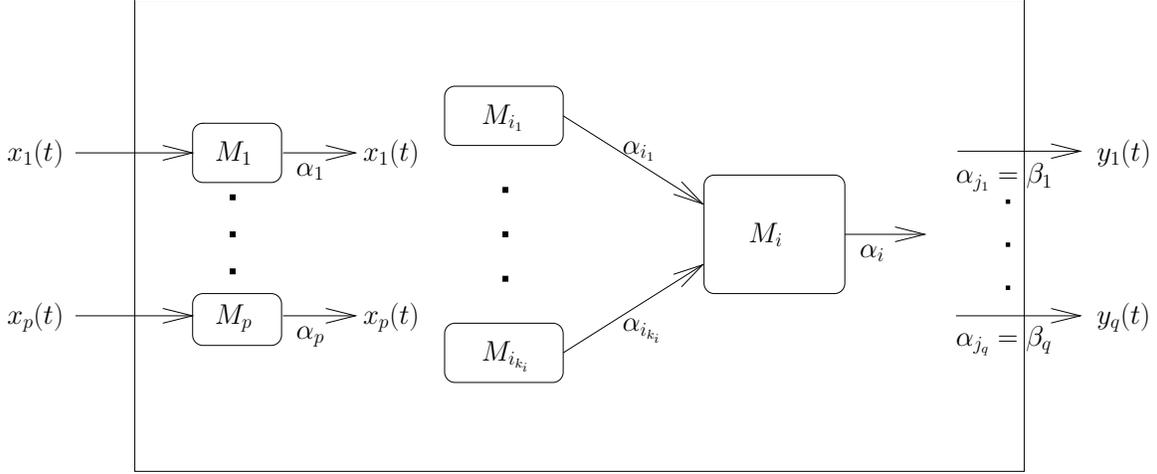


FIGURE 2: A network

We make an assumption of *input determinacy*:

InDet: For $i = 1, \dots, p$, the input channel α_i carries an input stream or signal $x_i(t)$ at all times $t \geq 0$.

2.4 Network operation: the model; well-posedness

Under the module function causality assumption **Caus**, we want to prove a *network determinacy* condition:

NetDet: For certain inputs and parameter values, there is a well-determined value for the stream on each channel at all times.

This means that, at least for a certain set $U \subseteq A^r \times \mathcal{C}[\mathbb{T}, A]^p$ of global parameters and stream inputs $(\mathbf{c}, \mathbf{x}) \in U$, there is a well-determined tuple of (total) streams

$$\mathbf{u} = (u_1, \dots, u_m) \in \mathcal{C}[\mathbb{T}, A]^m$$

that describes the data on all channels $\alpha_1, \dots, \alpha_m$ at all times. “Well-determinedness” of the tuple $\mathbf{u} = (u_1, \dots, u_m)$, or “well-posedness” of the problem, implies, further, *stability under perturbation*, i.e., *continuity* of the stream tuple \mathbf{u} as a function of the inputs $(\mathbf{c}, \mathbf{x}) \in U$ (cf. Remark 3.3.2 below). Thus with each module M_i ($i = 1, \dots, m$) is associated a continuous (partial) function

$$\Phi_i: A^r \times \mathcal{C}[\mathbb{T}, A]^p \rightarrow \mathcal{C}[\mathbb{T}, A]$$

where, for $(\mathbf{c}, \mathbf{x}) \in U$:

$$\Phi_i(\mathbf{c}, \mathbf{x}) = u_i.$$

These module functions Φ_i ($i = 1, \dots, m$) can then be vectorised to form the *network state function*

$$\Phi^N: A^r \times \mathcal{C}[\mathbb{T}, A]^p \rightarrow \mathcal{C}[\mathbb{T}, A]^m$$

where, for $(\mathbf{c}, \mathbf{x}) \in U$:

$$\Phi^N(\mathbf{c}, \mathbf{x}) = (\Phi_1(\mathbf{c}, \mathbf{x}), \dots, \Phi_m(\mathbf{c}, \mathbf{x})) \quad (2.1)$$

and hence also a continuous *network i/o function*

$$\Phi_{\text{io}}^N: A^r \times \mathcal{C}[\mathbb{T}, A]^p \rightarrow \mathcal{C}[\mathbb{T}, A]^q$$

where, for $(\mathbf{c}, \mathbf{x}) \in U$:

$$\Phi_{\text{io}}^N(\mathbf{c}, \mathbf{x}) = (\Phi_{j_1}(\mathbf{c}, \mathbf{x}), \dots, \Phi_{j_q}(\mathbf{c}, \mathbf{x})),$$

i.e., $\Phi_{\text{io}}^N(\mathbf{c}, \mathbf{x})$ is a suitable sub-tuple of $\Phi^N(\mathbf{c}, \mathbf{x})$, as we will see below (Theorem 2 and Remark 3.3.2).

2.5 Network operation: Algebraic specification

Given the above assumptions, we can specify the model by the following *system equations*:

$$u_i(t) = x_i(t) \quad (i = 1, \dots, p, \quad t \geq 0) \quad (2.2a)$$

$$u_i(t) = F_i(u_{i_1}, \dots, u_{i_{k_i}}, c_{i_1}, \dots, c_{i_{l_i}})(t) \quad (i = p + 1, \dots, m, \quad t \geq 0), \quad (2.2b)$$

which form an *algebraic specification* for the network state function to be constructed below (Section 3). Here (2.2a) is the *input condition*.

In Section 3 we will derive the existence and uniqueness of solutions of this specification as a fixed point of a certain function.

Remark 2.5.1. The specifying equations (2.2) include input conditions (2.2a) but not any “initialisation conditions” of the form $u_i(0) = \dots$, in contrast to the situation with SCAs (see Remark 2.2.4(b)). This is connected with the fact that SCAs are *underdetermined* without initial conditions, whereas our analog networks are fully determined by (2.2), assuming any solution exists.

In certain cases, initial values on some of the channels may in fact be given by the values of the module parameters, typically as constants of integration (as we will see in the case studies in Section 4); however such initial values are then determined through the corresponding equation (2.2a); they do not have to be specified by further “initialisation equations”.

3 Solving network equations; Fixed point semantics

We want to construct a *network state function*, which will be an m -tuple of *module state functions*, satisfying the equational algebraic specification (2.2).

First, we define some general concepts and give some results concerning the topology of stream spaces and stream transformations. More details can be found in [TZ07].

3.1 Stream spaces and stream transformations

Let $0 \leq a < b$, and let $\mathcal{C}[[a, b], A]$ be the set of continuous functions from $[a, b]$ to A . For $u, v \in \mathcal{C}[[a, b], A]$ (or $u, v \in \mathcal{C}[\mathbb{T}, A]$), define

$$\mathbf{d}_{a,b}(u, v) =_{df} \sup \{ \mathbf{d}_A(u(t), v(t)) \mid t \in [a, b] \}.$$

This makes $\mathcal{C}[[a, b], A]$ a complete metric space, with the *topology of uniform convergence*. The product space $\mathcal{C}[[a, b], A]^m$ (for $m > 0$) has the metric

$$\mathbf{d}_{a,b}^m(\mathbf{u}, \mathbf{v}) = \left(\sum_{i=1}^m (\mathbf{d}_{a,b}(u_i, v_i))^p \right)^{\frac{1}{p}} \quad (3.1)$$

(where $\mathbf{u} = (u_1, \dots, u_m)$ and $\mathbf{v} = (v_1, \dots, v_m)$) for some fixed p ($1 \leq p \leq \infty$). Two common special cases are formed by taking $p = 1$:

$$\mathbf{d}_{a,b}^m(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^m \mathbf{d}_{a,b}(u_i, v_i)$$

and $p = \infty$:

$$\mathbf{d}_{a,b}^m(\mathbf{u}, \mathbf{v}) = \max_{i=1}^m \mathbf{d}_{a,b}(u_i, v_i).$$

We will usually drop the superscript ‘ m ’ from $\mathbf{d}_{a,b}^m$.

Of special interest to us are the spaces $\mathcal{C}[[0, k], A]$ of streams on the finite intervals $[0, k]$ ($k = 1, 2, \dots$). We will write \mathbf{d}_k for the metric $\mathbf{d}_{0,k}$.

The *stream space* $\mathcal{C}[\mathbb{T}, A]$ cannot, in general, be made into a metric space, and $\mathbf{d}_{a,b}$ is only a pseudometric on $\mathcal{C}[\mathbb{T}, A]$. Nevertheless we can define a notion of convergence in $\mathcal{C}[\mathbb{T}, A]$ as follows.

Definition 3.1.1 (Local uniform convergence in $\mathcal{C}[\mathbb{T}, A]$). A sequence (u_n) of elements of $\mathcal{C}[\mathbb{T}, A]$ *converges locally uniformly* to the limit $u \in \mathcal{C}[\mathbb{T}, A]$ if

$$\forall \epsilon > 0 \forall k \exists N \forall n \geq N : \mathbf{d}_k(u_n, u) \leq \epsilon,$$

or, more simply but equivalently:

$$\forall k \exists N \forall n \geq N : \mathbf{d}_k(u_n, u) \leq 2^{-k}.$$

Such a limit (if it exists) is easily seen to be unique.

The *topology of local uniform convergence* can then be characterised as follows. Given a set $X \subseteq \mathcal{C}[\mathbb{T}, A]$ and a point $u \in \mathcal{C}[\mathbb{T}, A]$, u is in the *closure* of X if, and only if, there is a sequence of elements of X which converges locally uniformly to u . Equivalently, it is the topology generated by open neighbourhoods of the form

$$\mathbf{N}_k(u, r) =_{df} \{ v \in \mathcal{C}[\mathbb{T}, A] \mid \mathbf{d}_k(v, u) < r \}$$

for any stream u , any $r > 0$ and $k = 1, 2, \dots$.

This topology on $\mathcal{C}[\mathbb{T}, A]$ can also be characterised as the *inverse limit* [Eng89, §2.5] of the family of metric spaces and mappings

$$\lim_{\leftarrow k} \langle \mathcal{C}[[0, k], A], \pi_k \rangle$$

where the mapping $\pi_k: \mathcal{C}[[0, k+1], A] \rightarrow \mathcal{C}[[0, k], A]$ is defined by $\pi_k(u) = u \upharpoonright_k$. The projection mappings $\pi^k: \mathcal{C}[\mathbb{T}, A] \rightarrow \mathcal{C}[[0, k], A]$ are then given by $\pi^k(u) = u \upharpoonright_k$.

Remark 3.1.2 (Standard compact exhaustion of \mathbb{T}). We can think of the sequence of sets $[0, k]$ ($k = 0, 1, 2, \dots$) as a *compact exhaustion* (the “standard” one) of \mathbb{T} , which demonstrates the *sequential compactness* [TZ07, §2] of \mathbb{T} . Note that our characterisations of the topology of local uniform convergence and the inverse limit topology used the complete metric spaces $\mathcal{C}[K, A]$ where K has (only) the special form $[0, k]$ of this standard compact exhaustion. We could, alternatively, have let K range over $[a, b]$ for $0 \leq a < b$, or over arbitrary compact subsets of \mathbb{T} . These produce the same topology [TZ07]. Our choice gives the notationally simplest development. The same remarks apply to the exact form of the definition (3.1.3) of locally uniform Cauchy sequence below.

Finally, this topology is the same as the *compact-open topology* on $\mathcal{C}[\mathbb{T}, A]$ [Eng89, §3.4], which is defined as having *subbasic open sets* of the form

$$\mathbf{M}(K, U) =_{df} \{ u \in \mathcal{C}[\mathbb{T}, A] \mid \forall t \in K : u(t) \in U \}$$

for all compact $K \subset \mathbb{T}$ and open subsets U of A .

The equivalence of all the above characterisations of the topology on $\mathcal{C}[\mathbb{T}, A]$ is proved in [TZ07].

Moreover, the space $\mathcal{C}[\mathbb{T}, A]$ is *complete* in the sense given below (Lemma 3.1.4). First we define:

Definition 3.1.3 (Locally uniform Cauchy sequence). A sequence (u_n) of elements of $\mathcal{C}[\mathbb{T}, A]$ is a *locally uniform Cauchy sequence* if

$$\forall \epsilon > 0 \forall k \exists N \forall m, n \geq N : d_k(u_m, u_n) \leq \epsilon,$$

or, more simply but equivalently:

$$\forall k \exists N \forall m, n \geq N : d_k(u_m, u_n) \leq 2^{-k}.$$

Lemma 3.1.4 (Completeness of $\mathcal{C}[\mathbb{T}, A]$). A *locally uniform Cauchy sequence* in $\mathcal{C}[\mathbb{T}, A]$ converges *locally uniformly* to a limit.

Proof: Let (u_n) be a locally uniform Cauchy sequence in $\mathcal{C}[\mathbb{T}, A]$. For any k , the sequence $u_0 \upharpoonright_k, u_1 \upharpoonright_k, \dots$ is a uniform Cauchy sequence in the space $\mathcal{C}[[0, k], A]$, and so, by completeness of $\mathcal{C}[[0, k], A]$, has a limit $u^{(k)}$ in $\mathcal{C}[[0, k], A]$. These limits are *compatible*, in the sense

that for $n > k$, $u^{(k)} = u^{(n)} \upharpoonright_k$. The desired limit u can then be defined as the common extension of all the $u^{(k)}$, *i.e.*, $u(t) = u^{(k)}(t)$ for any $k \geq t$. \square

We are interested in stream transformations

$$f: \mathcal{C}[\mathbb{T}, A]^m \rightarrow \mathcal{C}[\mathbb{T}, A]^m \quad (m > 0). \quad (3.2)$$

We are especially interested in *contracting stream transformations*, to be explained below. First, some notation and definitions.

Notation 3.1.5. For $\mathbf{u} \in \mathcal{C}[\mathbb{T}, A]^m$ and $T > 0$, define $\mathbf{u} \upharpoonright_T \in \mathcal{C}[\mathbb{T}, A]^m$ by

$$\mathbf{u} \upharpoonright_T(t) = \begin{cases} \mathbf{u}(t) & \text{if } t \leq T \\ \mathbf{u}(T) & \text{if } t > T. \end{cases}$$

In other words, the stream $\mathbf{u} \upharpoonright_T$ agrees with the stream \mathbf{u} up to time T , and thereafter has, as constant value, the value of \mathbf{u} at T .

Remark 3.1.6 (Causality for stream transformations). For reasons that will become clear in the following development, we will generally assume that the stream transformations satisfy a causality condition (*cf.* Definition 2.2.3), which, for a stream transformation f of the form (3.2) can be most conveniently expressed as:

$$\mathbf{Caus}: \quad \text{For all } T \geq 0 \text{ and } \mathbf{u}, \mathbf{v} \in \mathcal{C}[\mathbb{T}, A]^m: \quad \mathbf{u} \upharpoonright_T = \mathbf{v} \upharpoonright_T \implies f(\mathbf{u}) \upharpoonright_T = f(\mathbf{v}) \upharpoonright_T.$$

Definition 3.1.7 (Contracting stream transformations). Let $0 < \lambda < 1$ and $\tau > 0$. A stream transformation f as in (3.2) is said to be *contracting w.r.t.* (λ, τ) , or to be in $\mathbf{Contr}(\lambda, \tau)$, if for all $T \geq 0$ and all $\mathbf{u}, \mathbf{v} \in \mathcal{C}[\mathbb{T}, A]^m$:

$$d_{T, T+\tau}(f(\mathbf{u}), f(\mathbf{v})) \leq \lambda \cdot d_{T, T+\tau}(\mathbf{u}, \mathbf{v}). \quad (3.3)$$

The factor λ is said to be a *modulus of contraction for f w.r.t. τ* .

Lemma 3.1.8. Suppose f satisfies \mathbf{Caus} . Then if $f \in \mathbf{Contr}(\lambda, \tau)$ for some $\tau > 0$, then $f \in \mathbf{Contr}(\lambda, \tau')$ for all $\tau' > 0$.

Proof: Suppose $f \in \mathbf{Contr}(\lambda, \tau)$ for a given τ . We must show that for any $\tau' > 0$, $f \in \mathbf{Contr}(\lambda, \tau')$. The proof is in three stages.

(i) First, $f \in \mathbf{Contr}(\lambda, k\tau)$ for any positive integer k . The proof does not depend on \mathbf{Caus} . For suppose

$$d_{T, T+k\tau}(\mathbf{u}, \mathbf{v}) = \delta.$$

Then clearly, for $j = 0, \dots, k-1$,

$$d_{T+j\tau, T+(j+1)\tau}(\mathbf{u}, \mathbf{v}) \leq \delta$$

and so, for $j = 0, \dots, k-1$, by **Contr** (λ, τ) ,

$$\mathbf{d}_{T+j\tau, T+(j+1)\tau}(f(\mathbf{u}), f(\mathbf{v})) \leq \lambda \cdot \delta$$

Hence

$$\mathbf{d}_{T, T+k\tau}(f(\mathbf{u}), f(\mathbf{v})) \leq \lambda \cdot \delta$$

from which it follows that $f \in \mathbf{Contr}(\lambda, k\tau)$.

(ii) Next, let $0 < \tau' < \tau$, and suppose

$$\mathbf{d}_{T, T+\tau'}(\mathbf{u}, \mathbf{v}) = \delta.$$

Then, using Notation 3.1.5:

$$\begin{aligned} \mathbf{d}_{T, T+\tau}(\mathbf{u} \upharpoonright_{T+\tau'}, \mathbf{v} \upharpoonright_{T+\tau'}) &= \sup_{t \in [T, T+\tau]} \mathbf{d}_A(\mathbf{u} \upharpoonright_{T+\tau'}(t), \mathbf{v} \upharpoonright_{T+\tau'}(t)) \\ &= \max\left(\sup_{t \in [T, T+\tau']} \mathbf{d}_A(\mathbf{u} \upharpoonright_{T+\tau'}(t), \mathbf{v} \upharpoonright_{T+\tau'}(t)), \right. \\ &\quad \left. \sup_{t \in [T+\tau', T+\tau]} \mathbf{d}_A(\mathbf{u} \upharpoonright_{T+\tau'}(t), \mathbf{v} \upharpoonright_{T+\tau'}(t)) \right) \\ &= \max\left(\sup_{t \in [T, T+\tau']} \mathbf{d}_A(\mathbf{u}(t), \mathbf{v}(t)), \right. \\ &\quad \left. \mathbf{d}_A(\mathbf{u}(T+\tau'), \mathbf{v}(T+\tau')) \right) \\ &= \sup_{t \in [T, T+\tau']} \mathbf{d}_A(\mathbf{u}(t), \mathbf{v}(t)), \\ &= \mathbf{d}_{T, T+\tau'}(\mathbf{u}, \mathbf{v}) \\ &= \delta. \end{aligned}$$

Hence, since $f \in \mathbf{Contr}(\lambda, \tau)$,

$$\mathbf{d}_{T, T+\tau}(f(\mathbf{u} \upharpoonright_{T+\tau'}), f(\mathbf{v} \upharpoonright_{T+\tau'})) \leq \lambda \cdot \delta. \quad (3.4)$$

Also, since $(\mathbf{u} \upharpoonright_{T+\tau'}) \upharpoonright_{T+\tau'} = \mathbf{u} \upharpoonright_{T+\tau'}$ and f satisfies **Caus**,

$$f(\mathbf{u} \upharpoonright_{T+\tau'}) \upharpoonright_{T+\tau'} = f(\mathbf{u}) \upharpoonright_{T+\tau'}$$

and similarly

$$f(\mathbf{v} \upharpoonright_{T+\tau'}) \upharpoonright_{T+\tau'} = f(\mathbf{v}) \upharpoonright_{T+\tau'}.$$

Hence

$$\begin{aligned} \mathbf{d}_{T, T+\tau'}(f(\mathbf{u}), f(\mathbf{v})) &= \mathbf{d}_{T, T+\tau'}(f(\mathbf{u} \upharpoonright_{T+\tau'}), f(\mathbf{v} \upharpoonright_{T+\tau'})) \\ &\leq \mathbf{d}_{T, T+\tau}(f(\mathbf{u} \upharpoonright_{T+\tau'}), f(\mathbf{v} \upharpoonright_{T+\tau'})) && \text{trivially} \\ &\leq \lambda \cdot \delta && \text{by (3.4)} \end{aligned}$$

Hence $f \in \mathbf{Contr}(\lambda, \tau')$.

(iii) Finally, for any $\tau' > 0$, $f \in \mathbf{Contr}(\lambda, \tau')$ follows by noting that $\tau' < k\tau$ for some k , and applying (i) and (ii). \square

Remark 3.1.9. A consequence of the above lemma is that if f satisfies **Caus** and $f \in \mathbf{Contr}(\lambda, \tau)$, then we can choose τ to suit ourselves. In such cases we will write $\mathbf{Contr}(\lambda)$ instead of $\mathbf{Contr}(\lambda, \tau)$, and generally take $\tau = 1$. We then say simply that f is *contracting w.r.t. λ* , and call λ a *modulus of contraction for f* .

The following theorem is fundamental in finding the solution of the network specifications.

Theorem 1 (Fixed point of contracting stream transformation). *Suppose the stream transformation f satisfies **Caus**, and $f \in \mathbf{Contr}(\lambda)$ for some $\lambda < 1$. Then f has a unique fixed point, i.e., there is a unique $\mathbf{u} \in \mathcal{C}[\mathbb{T}, A]^m$ such that $f(\mathbf{u}) = \mathbf{u}$.*

Proof: 1. Uniqueness:

Suppose \mathbf{u}, \mathbf{v} are fixed points of f . Then for all k

$$\begin{aligned} d_k(\mathbf{u}, \mathbf{v}) &= d_k(f(\mathbf{u}), f(\mathbf{v})) \\ &\leq \lambda \cdot d_k(\mathbf{u}, \mathbf{v}) \end{aligned}$$

since $f \in \mathbf{Contr}(\lambda)$, from which it follows that

$$d_k(\mathbf{u}, \mathbf{v}) = 0,$$

i.e.,

$$\mathbf{u} \upharpoonright_k = \mathbf{v} \upharpoonright_k$$

for all k , and hence

$$\mathbf{u} = \mathbf{v}.$$

2. Existence:

We will construct a solution, namely a fixed point \mathbf{v} of f , as a limit of a locally uniformly convergent Cauchy sequence of stream tuples:

$$\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \dots \tag{3.5}$$

Define \mathbf{v}_0 arbitrarily, and

$$\mathbf{v}_{n+1} = f(\mathbf{v}_n). \tag{3.6}$$

Then for all k and n , it can be seen, by induction on n , that

$$d_k(\mathbf{v}_n, \mathbf{v}_{n+1}) \leq \lambda^n d_k(\mathbf{v}_0, \mathbf{v}_1). \tag{3.7}$$

If $\mathbf{v}_1 = f(\mathbf{v}_0) = \mathbf{v}_0$, then \mathbf{v}_0 is the sought-for fixed point. Otherwise, $d_k(\mathbf{v}_0, \mathbf{v}_1) > 0$ for k sufficiently large. The sequence $(\mathbf{v}_n)_n$ can then be seen to be a locally uniform Cauchy sequence, by (given k and $\epsilon > 0$) choosing N (in Definition 3.1.1, 1st version) such that

$$\lambda^N < \frac{(1 - \lambda) \cdot \epsilon}{d_k(\mathbf{v}_0, \mathbf{v}_1)}. \tag{3.8}$$

For then, for $m > n \geq N$,

$$\begin{aligned}
d_k(\mathbf{v}_n, \mathbf{v}_m) &\leq d_k(\mathbf{v}_n, \mathbf{v}_{n+1}) + \dots + d_k(\mathbf{v}_{m-1}, \mathbf{v}_m) \\
&\leq (\lambda^n + \lambda^{n+1} + \dots + \lambda^m) \cdot d_k(\mathbf{v}_0, \mathbf{v}_1) && \text{by (3.7)} \\
&< (\lambda^n + \lambda^{n+1} + \dots + \lambda^m + \lambda^{m+1} + \dots) \cdot d_k(\mathbf{v}_0, \mathbf{v}_1) \\
&= \lambda^n \cdot (1 - \lambda)^{-1} d_k(\mathbf{v}_0, \mathbf{v}_1) \\
&\leq \lambda^N \cdot (1 - \lambda)^{-1} d_k(\mathbf{v}_0, \mathbf{v}_1) \\
&< \epsilon && \text{by (3.8)}.
\end{aligned}$$

Thus by Lemma 3.1.4 (which applies to $\mathcal{C}[\mathbb{T}, A]^m$ as well as $\mathcal{C}[\mathbb{T}, A]$) the sequence (3.5) converges locally uniformly to a limit \mathbf{v} .

Hence, also, the sequence

$$f(\mathbf{v}_0), f(\mathbf{v}_1), f(\mathbf{v}_2), \dots \quad (3.9)$$

converges locally uniformly to $f(\mathbf{v})$, since by the contraction property of f ,

$$d_k(f(\mathbf{v}_n), f(\mathbf{v})) \leq \lambda \cdot d_k(\mathbf{v}_n, \mathbf{v}).$$

for all k and n . Since (3.9) is actually the sequence (3.5) shifted by 1, it follows that it also converges to \mathbf{v} , and so

$$f(\mathbf{v}) = \mathbf{v}. \quad \square$$

Remark 3.1.10 (Contracting transformation w.r.t. compact exhaustion). Our definition (3.1.7) of contracting transformation uses a rather strong or “global” notion of contraction (3.3), holding for all $T > 0$. A more general definition [TZ07] has a weaker notion of contraction, formulated relative to the standard exhaustion of \mathbb{T} , in which the global contraction constant λ is replaced by a sequence $(\lambda_0, \lambda_1, \lambda_2, \dots)$ of contraction constants, each strictly between 0 and 1, and then (3.3) is replaced by

$$d_k(f(\mathbf{u}), f(\mathbf{v})) \leq \lambda_k \cdot d_k(\mathbf{u}, \mathbf{v}) \quad (k = 0, 1, 2, \dots).$$

The appropriate version of Theorem 1 can still be derived for this formulation of contraction.

Interestingly, for this formulation of the contraction property, we do not seem to need the *Caus* condition, which was needed in our proof of Theorem 1.

The notion of contraction that we used (3.3) is, however, simple to work with, and sufficient for the two case studies in Section 4.

Remark 3.1.11 (Effectivity of local uniform convergence). Note that in Section 5, where we deal with the issue of the computability of the fixed point \mathbf{u} , we need a stronger property of the sequence (3.5) than local uniform convergence, namely *effective* local uniform convergence.

We turn to apply the above theory to the network N .

3.2 Network functions

Recall the specifications for the network N in Section 2:

$$u_i(t) = x_i(t) \quad (i = 1, \dots, p, \quad t \geq 0) \quad (2.2a)$$

$$u_i(t) = F_i(u_{i1}, \dots, u_{ik_i}, c_{i1}, \dots, c_{il_i})(t) \quad (i = p+1, \dots, m, \quad t \geq 0), \quad (2.2b)$$

Writing the *global parameters* as $\mathbf{c} = (c_1, \dots, c_r)$, and the *input streams* as $\mathbf{x} = (x_1, \dots, x_p) \in \mathcal{C}[\mathbb{T}, A]^p$, a (partial) solution (u_1, \dots, u_m) to these equations is given by a subset $U \subseteq A^r \times \mathcal{C}[\mathbb{T}, A]^p$, and, for each module M_i ($i = 1, \dots, m$) a function

$$\Phi_i: A^r \times \mathcal{C}[\mathbb{T}, A]^p \rightarrow \mathcal{C}[\mathbb{T}, A]$$

where, for all $(\mathbf{c}, \mathbf{x}) \in U$:

$$\Phi_i(\mathbf{c}, \mathbf{x}) = u_i,$$

from which, as we have seen in Section 2, we obtain by vectorisation the network state function for N :

$$\Phi^N: A^r \times \mathcal{C}[\mathbb{T}, A]^p \rightarrow \mathcal{C}[\mathbb{T}, A]^m,$$

where for $(\mathbf{c}, \mathbf{x}) \in U$:

$$\Phi^N(\mathbf{c}, \mathbf{x}) = (\Phi_1(\mathbf{c}, \mathbf{x}), \dots, \Phi_m(\mathbf{c}, \mathbf{x})). \quad (2.1)$$

Notice next that a stream tuple (u_1, \dots, u_m) satisfying the specifications (2.2) can be written as

$$(u_1, \dots, u_m) = (x_1, \dots, x_p, u_{p+1}^0, \dots, u_m^0)$$

where $\mathbf{x} = (x_1, \dots, x_p)$ are the input streams of N , and $\mathbf{u}^0 = (u_{p+1}^0, \dots, u_m^0)$ form a *fixed point* of the *network stream transformation* function

$$\Psi_{\mathbf{c}, \mathbf{x}}^N: \mathcal{C}[\mathbb{T}, A]^{m-p} \rightarrow \mathcal{C}[\mathbb{T}, A]^{m-p}$$

with

$$\Psi_{\mathbf{c}, \mathbf{x}}^N(\mathbf{u}) =_{df} \Psi^N(\mathbf{c}, \mathbf{x}, \mathbf{u}), \quad (3.10)$$

where the function

$$\Psi^N: A^r \times \mathcal{C}[\mathbb{T}, A]^p \times \mathcal{C}[\mathbb{T}, A]^{m-p} \rightarrow \mathcal{C}[\mathbb{T}, A]^{m-p}$$

is defined by

$$\Psi^N(\mathbf{c}, \mathbf{x}, \mathbf{u}) =_{df} (F_{p+1}(\mathbf{u}_1, \mathbf{c}_1), \dots, F_m(\mathbf{u}_m, \mathbf{c}_m)) \quad (3.11)$$

where, on the r.h.s., $(\mathbf{u}_i, \mathbf{c}_i)$ is the list of local input streams and local parameters associated with F_i , with \mathbf{u}_i a sub-tuple of (\mathbf{x}, \mathbf{u}) , for $i = p+1, \dots, m$. (Recall that the operations of the modules F_1, \dots, F_p are just the identity functions.) Then

a solution to (2.2) will be a fixed point for $\Psi_{\mathbf{c}, \mathbf{x}}^N$.

The network state function Φ^N is then easily obtained from this fixed point \mathbf{u}^0 , since for a given *input* $(\mathbf{c}, \mathbf{x}) \in U$, the *output* of Φ^N is just $(\mathbf{x}, \mathbf{u}^0)$.

Thus in looking for a solution to the equations (2.2), the basic questions are:

Under what conditions does $\Psi_{\mathbf{c}, \mathbf{x}}^N$ have a fixed point?
Under what conditions is it unique?

We will give at least a partial solution to this, namely a sufficient condition for a fixed point, which will also be unique, by applying the theory of contracting stream transformations developed above.

3.3 Solution of fixed point equation

Recall Definition 3.1.7 and Remark 3.1.9.

Definition 3.3.1 (Contracting condition for a network).

Given $\mathbf{c} \in A^r$, $\mathbf{x} \in \mathcal{C}[\mathbb{T}, A]^p$ and $0 < \lambda < 1$, the network N satisfies $\mathbf{Contr}_{\mathbf{c}, \mathbf{x}}(\lambda)$ if the network stream transformation

$$\Psi_{\mathbf{c}, \mathbf{x}}^N: \mathcal{C}[\mathbb{T}, A]^{m-p} \rightarrow \mathcal{C}[\mathbb{T}, A]^{m-p}$$

defined by (3.10) and (3.11) is (total and) in $\mathbf{Contr}(\lambda)$. It is *contracting at* (\mathbf{c}, \mathbf{x}) if it satisfies $\mathbf{Contr}_{\mathbf{c}, \mathbf{x}}(\lambda)$ for some $\lambda < 1$.

Theorem 2. (a) (**Existence and Uniqueness**) Suppose that for all $(\mathbf{c}, \mathbf{x}) \in U \subseteq A^r \times \mathcal{C}[\mathbb{T}, A]^p$, there exists $\lambda (= \lambda_{\mathbf{c}, \mathbf{x}}) < 1$ such that the network N satisfies $\mathbf{Contr}_{\mathbf{c}, \mathbf{x}}(\lambda)$. Then there is a unique stream tuple $(u_1, \dots, u_m) \in \mathcal{C}[\mathbb{T}, A]^m$ satisfying the network equations (2.2). It is given by specifying that $(u_1, \dots, u_p) = \mathbf{x}$ and $(u_{p+1}, \dots, u_m) = \mathbf{u}^0$ is the unique fixed point of the stream transformation function $\Psi_{\mathbf{c}, \mathbf{x}}^N: \mathcal{C}[\mathbb{T}, A]^{m-p} \rightarrow \mathcal{C}[\mathbb{T}, A]^{m-p}$ defined by equations (3.10) and (3.11). This defines the network state function $\Phi^N: A^r \times \mathcal{C}[\mathbb{T}, A]^p \rightarrow \mathcal{C}[\mathbb{T}, A]^m$ and network i/o function $\Phi_{\text{i/o}}^N: A^r \times \mathcal{C}[\mathbb{T}, A]^p \rightarrow \mathcal{C}[\mathbb{T}, A]^q$ by: $\Phi^N(\mathbf{c}, \mathbf{x}) = (\mathbf{x}, \mathbf{u}^0)$, and $\Phi_{\text{i/o}}^N(\mathbf{c}, \mathbf{x})$ is a suitable sub-tuple of this, for all $(\mathbf{c}, \mathbf{x}) \in U$.

(b) (**Continuity**) Suppose further that the module functions of N are continuous, and for some (\mathbf{c}, \mathbf{x}) in the interior of U , the modulus of contraction λ can be defined in a neighbourhood of (\mathbf{c}, \mathbf{x}) so as to be continuous at (\mathbf{c}, \mathbf{x}) . Then Φ^N and $\Phi_{\text{i/o}}^N$ are continuous at (\mathbf{c}, \mathbf{x}) .

Proof: Part (a) is immediate from Theorem 1. For part (b): First note that we can assume without loss of generality that λ can be defined so as to be *constant* in a neighbourhood of (\mathbf{c}, \mathbf{x}) . For if λ is continuous at (\mathbf{c}, \mathbf{x}) , with value $\lambda_0 < 1$ at (\mathbf{c}, \mathbf{x}) , then, by continuity, its value is less than (say) $\lambda_1 =_{df} (\lambda_0 + 1)/2 < 1$ in some neighbourhood of (\mathbf{c}, \mathbf{x}) . So we can take the constant value λ_1 as the modulus of contraction near (\mathbf{c}, \mathbf{x}) .

We use the notation of Theorem 1 and its proof, notably (3.5) and (3.6). So, putting $f = \Psi_{\mathbf{c}, \mathbf{x}}^N$ (cf. (3.10), (3.11)), and \mathbf{v}_0 arbitrary but fixed, we have (cf. (3.5), (3.6)) $\mathbf{v}_n = f^{(n)}(\mathbf{v}_0)$ ($n = 0, 1, 2, \dots$), and so $\Phi^N(\mathbf{c}, \mathbf{x}) = (\mathbf{x}, \mathbf{u}^0)$ where \mathbf{u}^0 is the limit of the Cauchy sequence (\mathbf{v}_n) , which is a fixed point of f . Similarly, for each $(\mathbf{c}', \mathbf{x}') \in V$, writing

$f' = \Psi_{\mathbf{c}', \mathbf{x}'}^N$, and $\mathbf{v}'_n = f'^{(n)}(\mathbf{v}_0)$, we have $\Phi^N(\mathbf{c}', \mathbf{x}') = (\mathbf{x}', \mathbf{u}^{\mathbf{0}'})$ where $\mathbf{u}^{\mathbf{0}'}$ is the limit of the Cauchy sequence (\mathbf{v}'_n) , which is the fixed point of f' .

Hence to show that Φ^N is continuous at (\mathbf{c}, \mathbf{x}) , we must show that for $(\mathbf{c}', \mathbf{x}')$ sufficiently “close to” (\mathbf{c}, \mathbf{x}) , $\mathbf{u}^{\mathbf{0}'}$ is “close to” $\mathbf{u}^{\mathbf{0}}$.

Note that the product topology on $A^r \times \mathcal{C}[\mathbb{T}, A]^p$ is generated by the pseudometrics (cf. §3.1)

$$\mathbf{d}_k((\mathbf{c}, \mathbf{x}), (\mathbf{c}', \mathbf{x}')) = \max(\mathbf{d}_A(\mathbf{c}, \mathbf{c}'), \mathbf{d}_k(\mathbf{x}, \mathbf{x}'))$$

and the corresponding neighbourhoods

$$\mathbf{N}_k((\mathbf{c}, \mathbf{x}), r) =_{df} \{(\mathbf{c}', \mathbf{x}') \in A^r \times \mathcal{C}[\mathbb{T}, A]^p \mid \mathbf{d}_k((\mathbf{c}', \mathbf{x}'), (\mathbf{c}, \mathbf{x})) < r\}.$$

Since the module functions of N are continuous, Ψ^N is continuous, and so for all n , \mathbf{v}_n depends continuously on (\mathbf{c}, \mathbf{x}) . Now, given k and $\epsilon > 0$, choose n such that (cf. (3.8))

$$\lambda^n < \frac{(1 - \lambda) \cdot \epsilon}{6 \cdot \mathbf{d}_k(\mathbf{v}_0, \mathbf{v}_1)}. \quad (3.12)$$

Now choose $\delta > 0$ such that (i) $\mathbf{N}_k((\mathbf{c}, \mathbf{x}), \delta) \subseteq V$, and also (ii) the modulus of contraction has a constant value λ in $\mathbf{N}_k((\mathbf{c}, \mathbf{x}), \delta)$, and also (iii) for all $(\mathbf{c}', \mathbf{x}') \in \mathbf{N}_k((\mathbf{c}, \mathbf{x}), \delta)$,

$$\mathbf{d}_k(\mathbf{v}'_n, \mathbf{v}_n) < \epsilon/3, \quad (3.13)$$

and (iv) $\mathbf{d}_k(\mathbf{v}'_1, \mathbf{v}_1) < \mathbf{d}_k(\mathbf{v}_0, \mathbf{v}_1)$, so that

$$\begin{aligned} \mathbf{d}_k(\mathbf{v}_0, \mathbf{v}'_1) &\leq \mathbf{d}_k(\mathbf{v}_0, \mathbf{v}_1) + \mathbf{d}_k(\mathbf{v}_1, \mathbf{v}'_1) \\ &< 2 \cdot \mathbf{d}_k(\mathbf{v}_0, \mathbf{v}_1) \end{aligned} \quad (3.14)$$

Then (as in the proof of Theorem 1) for all $m > n$,

$$\begin{aligned} \mathbf{d}_k(\mathbf{v}_n, \mathbf{v}_m) &\leq \mathbf{d}_k(\mathbf{v}_n, \mathbf{v}_{n+1}) + \dots + \mathbf{d}_k(\mathbf{v}_{m-1}, \mathbf{v}_m) \\ &\leq (\lambda^n + \lambda^{n+1} + \dots + \lambda^m) \cdot \mathbf{d}_k(\mathbf{v}_0, \mathbf{v}_1) && \text{by (3.7)} \\ &< \lambda^n \cdot (1 - \lambda)^{-1} \cdot \mathbf{d}_k(\mathbf{v}_0, \mathbf{v}_1) \\ &< \epsilon/6 && \text{by (3.12)} \end{aligned}$$

and so

$$\mathbf{d}_k(\mathbf{v}_n, \mathbf{u}^{\mathbf{0}}) \leq \epsilon/6 < \epsilon/3. \quad (3.15)$$

Similarly, for all $m > n$,

$$\begin{aligned} \mathbf{d}_k(\mathbf{v}'_n, \mathbf{v}'_m) &< \lambda^n \cdot (1 - \lambda)^{-1} \cdot \mathbf{d}_k(\mathbf{v}_0, \mathbf{v}'_1) \\ &< \lambda^n \cdot (1 - \lambda)^{-1} \cdot 2 \cdot \mathbf{d}_k(\mathbf{v}_0, \mathbf{v}_1) && \text{by (3.14)} \\ &< \epsilon/3 && \text{by (3.12)} \end{aligned}$$

and so

$$\mathbf{d}_k(\mathbf{v}'_n, \mathbf{u}^{0'}) \leq \epsilon/3. \quad (3.16)$$

Hence

$$\begin{aligned} \mathbf{d}_k(\mathbf{u}^{0'}, \mathbf{u}^0) &\leq \mathbf{d}_k(\mathbf{v}', \mathbf{v}'_n) + \mathbf{d}_k(\mathbf{v}'_n, \mathbf{v}_n) + \mathbf{d}_k(\mathbf{v}_n, \mathbf{v}) \\ &< \epsilon/3 + \epsilon/3 + \epsilon/3 && \text{by (3.16), (3.13) and (3.15)} \\ &= \epsilon, \end{aligned}$$

proving the continuity of Φ^N at (\mathbf{c}, \mathbf{x}) . The continuity of Φ_{io}^N at (\mathbf{c}, \mathbf{x}) follows immediately, since the output of Φ_{io}^N at a given input is a sub-tuple of the output of Φ^N at the same input. \square

Remark 3.3.2 (Continuity, stability, well-posedness; Hadamard’s principle).

Continuity of the network state function, guaranteed by part (b) of the theorem under the stated conditions, implies in turn *stability* of the solution to the specification (2.2), and hence *well-posedness* of the problem. The significance of these issues is related to Hadamard’s principle [Had52] which, as (re-)formulated by Courant and Hilbert ([CH53, pp. 227ff.], [Had64]) states that for a scientific problem to be well posed, the solution must (apart from existing and being unique) depend continuously on the data.³

3.4 A simple example

Figure 3 shows a simple feedback system. Here the metric space A is \mathbb{R} . There are 4 channels. The *input channel* α_1 carries the *input stream* $x(t)$, and the *output channel* α_3 (also called β in conformity with the notation in Section 2) carries the *output stream* $y(t)$. There are 3 modules, M_1 , M_2 and M_3 . Module M_1 is the identity on the input stream x , M_2 sums the streams x and u to produce the output y and M_3 multiplies y by a scalar ρ to produce the stream u .

To simplify the treatment, we combine the scalar multiplier M_3 with the following module M_2 , to produce the network of Figure 4. (Henceforth we will not explicitly show the names α_i of the channels.)

We want to show that, for suitable values of the parameter ρ and input stream x , the network satisfies $\mathbf{Contr}_{\rho,x}(\lambda)$, *i.e.*, the function $\Psi_{\rho,x}$ of equation (3.11) is in $\mathbf{Contr}(\lambda)$ for some $\lambda < 1$.

Now the function F_2 associated with module M_2 is the “modified adder”

$$F_2(x, y) = x + \rho \cdot y.$$

The stream tuple \mathbf{u} of (3.12) is just the stream y . So for any $y \in \mathcal{C}[\mathbb{T}, \mathbb{R}]$,

$$\Psi_{\rho,x}(y) = F_2(x, y) = x + \rho \cdot y,$$

³See the insightful discussions on this topic by Beeson [Bee85, p. 368] and Myrvold [Myr95]

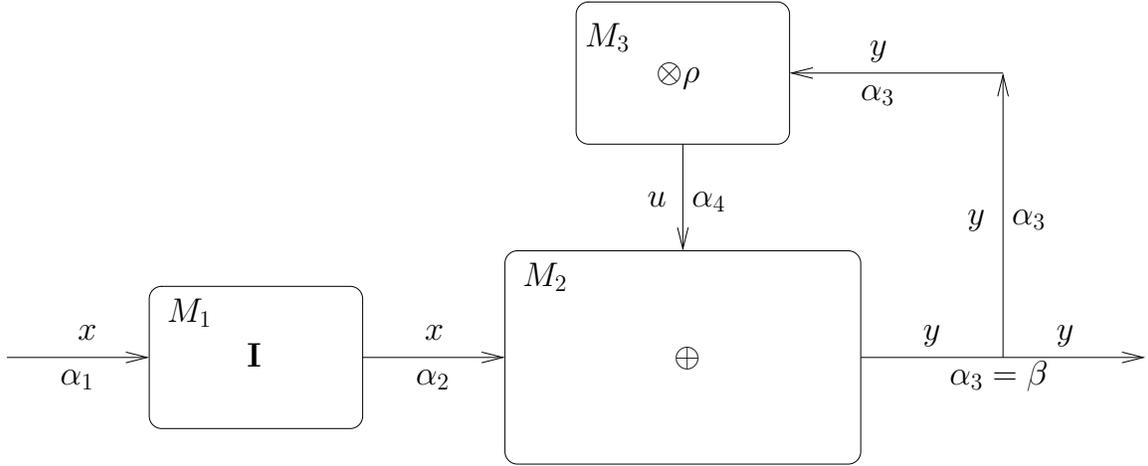


FIGURE 3: A simple feedback system

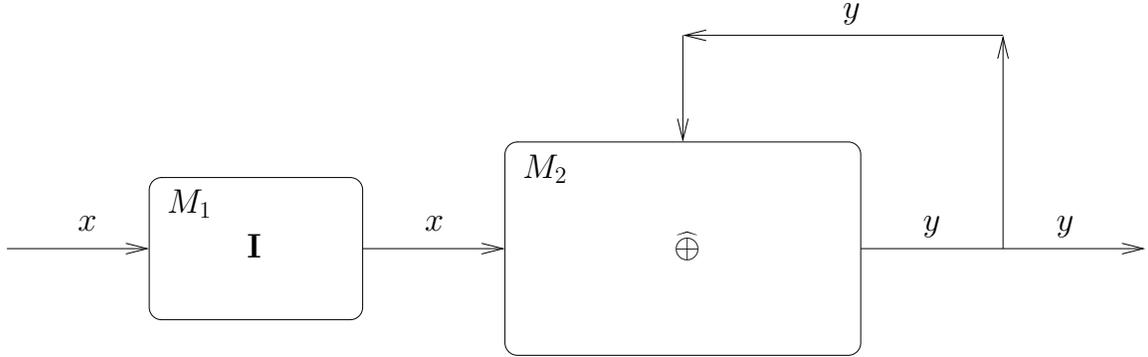


FIGURE 4: The feedback system simplified

or rather, for all $t \geq 0$:

$$\Psi_{\rho,x}(y)(t) = x(t) + \rho \cdot y(t).$$

We want a fixed point for $\Psi_{\rho,x}$, *i.e.*, a solution $y(t)$ to the equation

$$y(t) = x(t) + \rho \cdot y(t) \tag{3.17}$$

Note that for any $y_1, y_2 \in \mathcal{C}[\mathbb{T}, \mathbb{R}]$,

$$\Psi_{\rho,x}(y_1)(t) - \Psi_{\rho,x}(y_2)(t) = \rho \cdot (y_1(t) - y_2(t))$$

and so

$$|\Psi_{\rho,x}(y_1)(t) - \Psi_{\rho,x}(y_2)(t)| = \rho \cdot |y_1(t) - y_2(t)|$$

Hence for any T and $\tau > 0$:

$$\mathbf{d}_{T, T+\tau}(\Psi_{\rho,x}(y_1), \Psi_{\rho,x}(y_2)) = \rho \cdot \mathbf{d}_{T, T+\tau}(y_1, y_2).$$

Thus, if $\rho < 1$, this network satisfies **Contr**(ρ), and so has a solution y as output, for any input stream $x \in \mathcal{C}[\mathbb{T}, \mathbb{R}]$.

We can compute the output stream y , as a function of x , by the construction in the proof of Theorem 1, as follows. Define $y_0(t)$ arbitrarily, and

$$y_{n+1}(t) = \Psi_{\rho,x}(y_n)(t) = x(t) + \rho \cdot y_n(t).$$

It is easy to check that

$$\begin{aligned} y_{n+1}(t) &= (1 + \rho + \rho^2 + \cdots + \rho^n)x(t) + \rho^{n+1} \cdot y_0(t) \\ &= \frac{1 - \rho^{n+1}}{1 - \rho} x(t) + \rho^{n+1} \cdot y_0(t). \end{aligned}$$

The sequence $(y_n)_n$ is easily seen to be a locally uniform Cauchy sequence, with the limit

$$y(t) = \frac{1}{1 - \rho} x(t).$$

which is the required solution. Note that this limit is independent of the choice of y_0 . Also, it could have been obtained more easily (at least in this simple example, with one input and one output stream) by directly solving equation (3.17) for $y(t)$. Such a direct solution does not seem possible in more complicated examples, such as those in Section 4.

4 Two case studies

We apply the theory of Section 3 to two examples of analog designs taken from a standard text [Hyn70]. Each physical system is specified by a differential equation. To make an analog machine, we reconstruct the equation from its components, modelling it as a network of modules of the form described in Section 2. This process is relatively straightforward, though care must be taken to decompose the system appropriately so that it satisfies the contraction condition. Typically these modules are among the classical processing units such as scalar multipliers, integrators, etc. From this network an analog machine can be built.

4.1 Case study 1

(a) **Physical system:** The first case study (Figure 5) is a simple mass/spring/damper system, where a mass M is suspended by a spring with stiffness K and damping coefficient D . A force f (which is a function of time t) is applied to the mass. We want to compute its displacement x as a function of t .

(b) **Equational specification:** To set up the equation of motion, consider the three forces acting on the mass M : the external force f , the spring force $-Kx$, and the damping force $-Ddx/dt$.

By Newton's second law of motion:

$$Ma + Dv + Kx = f \tag{4.1}$$

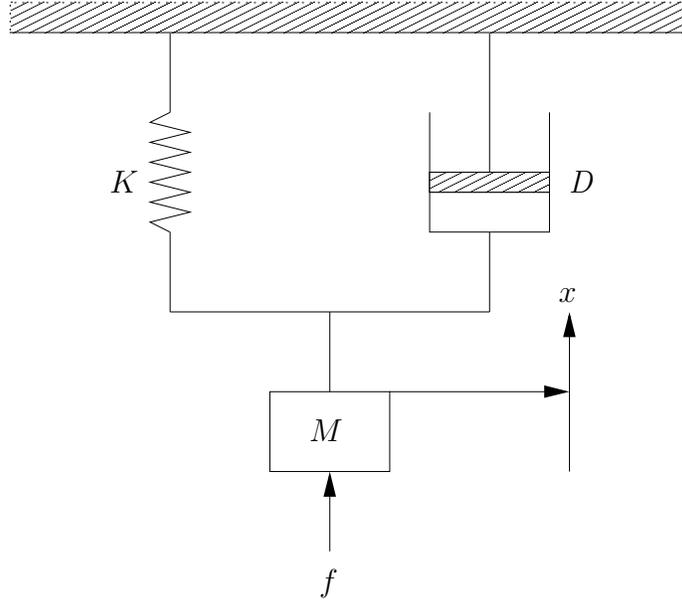


FIGURE 5: Case Study 1

where $v = dx/dt$ is the velocity of the mass, and $a = dv/dt$ is its acceleration.

We show how to design an analog machine that solves this equation for $x(t)$. The machine is designed to give the displacement $x(t)$ as a function of t by experiment, *i.e.*, measurements on the machine.

(c) **Network:** The analog network N for this system is shown in Figure 6. The *parameters* are M, K, D, v_0, x_0 (the first three used in module M_2 and the last two being constants of integration in M_3 and M_4 respectively), the single *input stream* is $f(t)$, and the single *output stream* is $x(t)$. The network follows [Hyn70], except for the extra “identity” module M_1 for the input stream f , as explained in Section 2. There are also an adder M_2 , integrators M_3 and M_4 , and scalar multipliers M_5, M_6 and M_7 .

Next we simplify the network by combining each scalar multiplier with the preceding or following module, as shown in Figure 7.

There are now 4 modules, M_1, \dots, M_4 . Then (recalling that F_i is the function computed by M_i) F_1 is the identity, $F_2 = \hat{\oplus}$ is the “modified adder”

$$a = F_2(f, x, v) = \frac{f - Kx - Dv}{M}$$

(obtained by rearranging (4.1)) and F_3 and F_4 are integrators:

$$\begin{aligned} v(t) &= F_3(a)(t) = \left(\int_0^t a \right) + v_0 \\ x(t) &= F_4(v)(t) = \left(\int_0^t v \right) + x_0 \end{aligned}$$

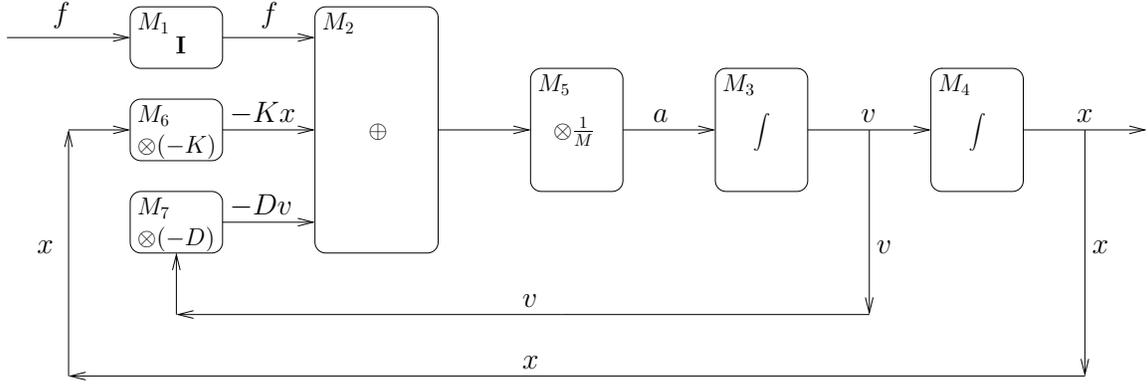


FIGURE 6: Network for case study 1

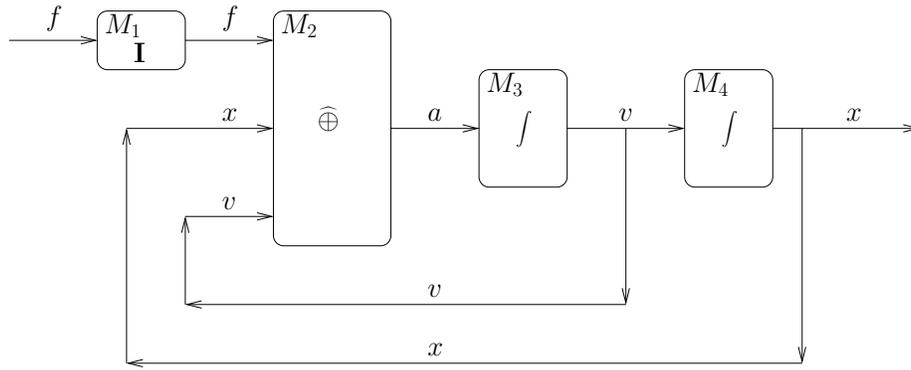


FIGURE 7: Simplified network for Case Study 1

Here the constants of integration v_0 and x_0 represent the initial velocity and displacement of the mass.

(d) **Network semantics:** The parameter tuple is

$$\mathbf{c} = (M, K, D, v_0, x_0),$$

the single *input stream* is f , and the *non-input stream tuple* is

$$\mathbf{u} = (a, v, x). \quad (4.2)$$

So we want a fixed point of the function

$$\Psi_{\mathbf{c}, f}: \mathcal{C}[\mathbb{T}, \mathbb{R}]^3 \rightarrow \mathcal{C}[\mathbb{T}, \mathbb{R}]^3$$

where

$$\Psi_{\mathbf{c}, f}(a, v, x) = (a', v', x')$$

with

$$a'(t) = \frac{1}{M}(f(t) - Kx(t) - Dv(t)) \quad (4.3a)$$

$$v'(t) = \int_0^t a(s)ds + v_0 \quad (4.3b)$$

$$x'(t) = \int_0^t v(s)ds + x_0. \quad (4.3c)$$

Now we must find under what conditions, $\Psi_{c,f}$ is contracting.

For changes $\delta a, \delta v, \delta x$ in a, v, x , and corresponding changes $\delta a', \dots$ in a', \dots :

$$\Psi_{c,f}(a + \delta a, v + \delta v, x + \delta x) = (a' + \delta a', v' + \delta v', x' + \delta x').$$

Now from (4.3a):

$$\delta a' = -\frac{1}{M}[K\delta x + D\delta v]$$

Hence for any $T \geq 0$ and $\tau > 0$, using the pseudonorm

$$\|u\| = \|u\|_T^{T+\tau} =_{df} \sup \{u(t) \mid T \leq t \leq T + \tau\} \quad (4.4)$$

we have from (4.3):

$$\|\delta a'\| \leq (K\|\delta x\| + D\|\delta v\|)/M \quad (4.5a)$$

$$\|\delta v'\| \leq \tau\|\delta a\| \quad (4.5b)$$

$$\|\delta x'\| \leq \tau\|\delta v\|. \quad (4.5c)$$

Now assume

$$M > \max(K, 2D) \quad (4.6)$$

and put

$$\lambda =_{df} \frac{\max(K, 2D)}{M}. \quad (4.7)$$

Then by (4.6)

$$\lambda < 1. \quad (4.8)$$

Let τ have any positive value $\leq D/M$, say

$$\tau = \frac{D}{M}. \quad (4.9)$$

Define the product pseudonorm

$$\|(\delta a, \delta v, \delta x)\| =_{df} \|\delta a\| + \|\delta v\| + \|\delta x\|. \quad (4.10)$$

This corresponds to the “sum definition” of the product pseudometric (equation (3.1), with $p = 1$). Then

$$\begin{aligned}
\|(\delta a', \delta v', \delta x')\| &= \|\delta a'\| + \|\delta v'\| + \|\delta x'\| \\
&\leq \frac{K}{M}\|\delta x\| + \left(\frac{D}{M} + \tau\right)\|\delta v\| + \tau\|\delta a\| && \text{by (4.5)} \\
&= \frac{K}{M}\|\delta x\| + \frac{2D}{M}\|\delta v\| + \frac{D}{M}\|\delta a\| && \text{by (4.9)} \\
&\leq \lambda\|(\delta a, \delta v, \delta x)\| && \text{by (4.7)}
\end{aligned}$$

which, by (4.8), proves $\mathbf{Contr}_{c,f}(\lambda)$.

Remark 4.1.1. The above calculation “worked” because we chose the “sum definition” (4.10) for the product pseudonorm. For some other example, another definition might work, with a different choice of p , or even a “mixed definition”, *e.g.*,

$$\|(u, v, w)\| =_{df} \max(\|u\|, \|v\|) + \|w\|.$$

Remark 4.1.2 (Effect of parameters). As it turned out, the only assumption needed to prove the contraction property was (4.6), *i.e.*, that the mass M be sufficiently large relative to the stiffness K and damping coefficient D . No assumption was needed on either the initial values v_0 and x_0 of velocity and displacement, or the external force $f(t)$. These remarks can be formulated as a theorem.

Theorem 3. *The network of Figure 7 is contracting, and hence satisfies \mathbf{NetDet} , for any input stream $f(t)$, provided $M > \max(K, 2D)$.*

Corollary 4.1.3. *The system of Figure 5 has a well determined solution $(a(t), v(t), x(t))$ for the acceleration, velocity and displacement as functions of time $t \geq 0$, for any input force $f(t)$ as a continuous function of time $t \geq 0$, and any initial conditions (v_0, x_0) for the velocity and displacement, provided only that $M > \max(K, 2D)$.*

4.2 Case Study 2: Coupled system

(a) **Physical system:** The second case studies, shown in Figure 8, is formed by coupling two mass/spring/damper systems. Now we want to compute the displacements x_1 and x_2 of both masses M_1 and M_2 . We will follow a method similar to that for Case Study 1.

(b) **Equational specification:** Equating the forces acting on each mass, we get the equations:

$$\begin{aligned}
M_1 a_1 + D_1 v_1 + K_1(x_1 - x_2) &= F(t) \\
M_2 a_2 + D_2 v_2 + K_2 x_2 + K_1(x_2 - x_1) &= 0
\end{aligned} \tag{4.11}$$

where v_i and a_i are the velocity and acceleration of a_i ($i = 1, 2$).

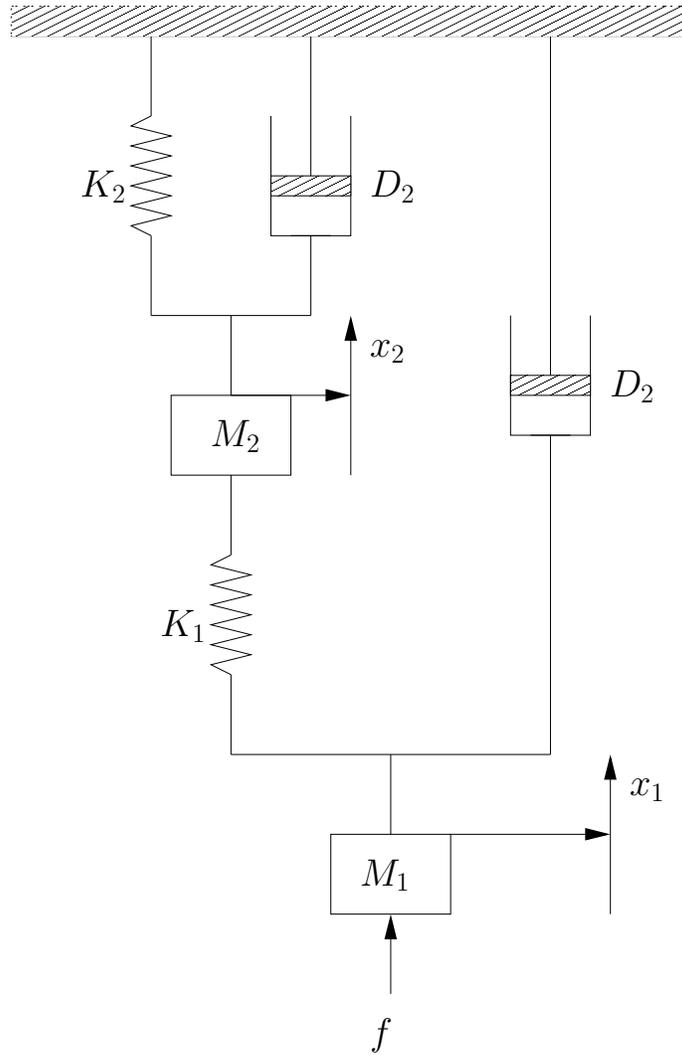


FIGURE 8: Case Study 2

(c) **Network:** The analog network N for this is shown in Figure 9. It is convenient to present the network as two components (reflecting Figure 8): an $M_1/K_1/D_1$ component and an $M_2/K_2/D_2$ component. In each component simplifications have been performed, similar to those used in transforming Figure 6 to Figure 7, *i.e.*, combining scalar multipliers with the preceding or following modules.

The *parameters* are $M_1, K_1, D_1, M_2, K_2, D_2, v_1^0, x_1^0, v_2^0, x_2^0$, the first three used in module M_{12} , the next three in module M_{13} , and the last four being constants of integration in $M_{13}, M_{14}, M_{23}, M_{24}$ respectively. There is one *input* stream, the external force $f(t)$, and two *output* streams, the displacements x_1 and x_2 of the two masses M_1 and M_2 .

There are now 7 modules M_{11}, \dots, M_{14} and M_{22}, \dots, M_{24} with corresponding module functions F_{ij} , where F_{11} is the identity, F_{12} and F_{22} are “modified adders” producing the

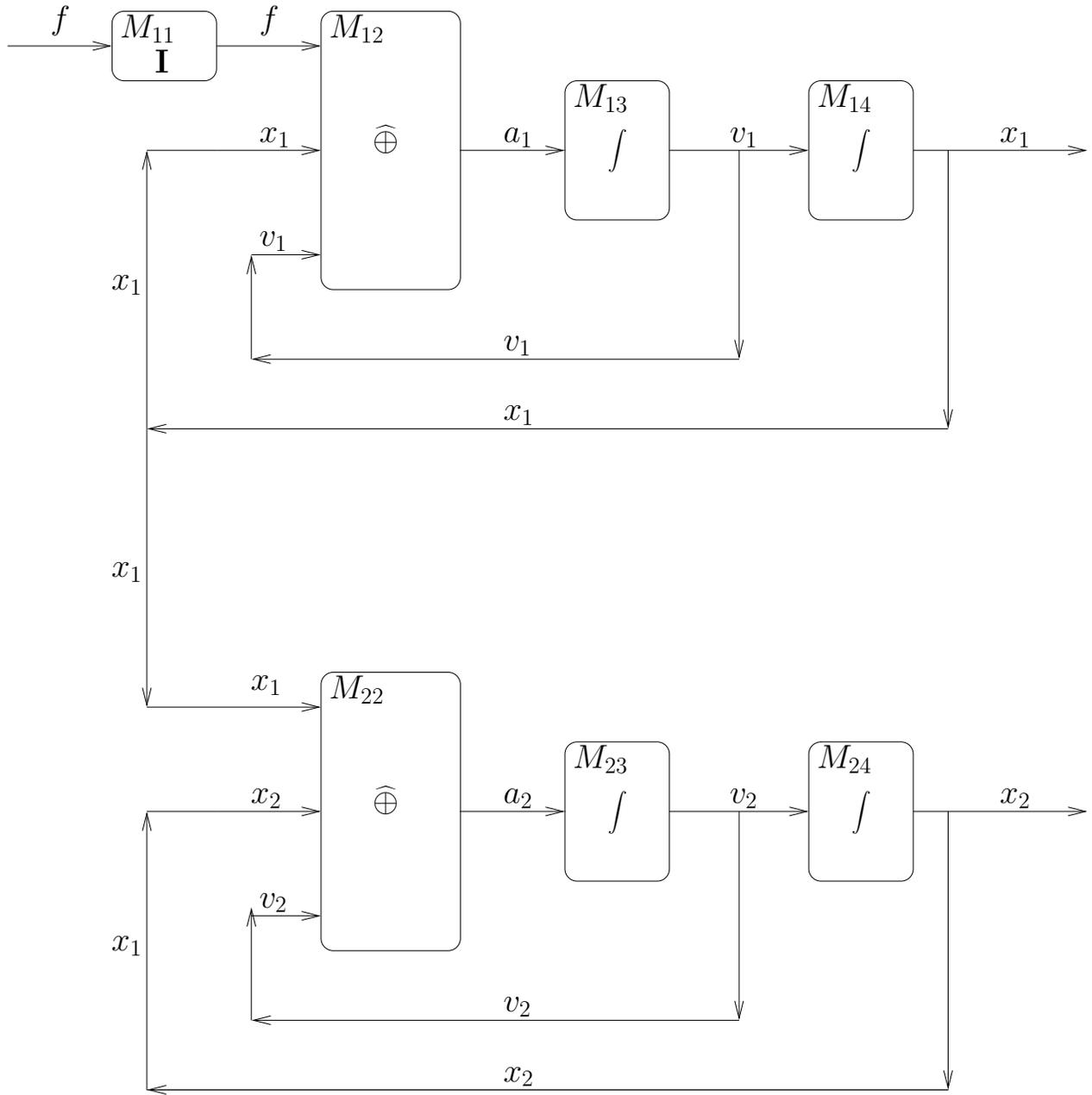


FIGURE 9: Simplified network for Case Study 2

two accelerations:

$$a_1 = F_{12}(f, x_1, v_1) = \frac{f - K_1(x_1 - x_2) - D_1 v_1}{M_1}$$

$$a_2 = F_{22}(x_1, x_2, v_2) = \frac{K_1(x_1 - x_2) - K_2 x_2 - D_2 v_2}{M_2}$$

(obtained by rearranging equations (4.11)) and $F_{13}, F_{14}, F_{23}, F_{24}$ are all integrators:

$$\begin{aligned} v_1(t) &= F_{13}(a)(t) = \int_0^t a_1(s)ds + v_1^0 \\ x_1(t) &= F_{14}(a)(t) = \int_0^t v_1(s)ds + x_1^0 \\ v_2(t) &= F_{23}(a)(t) = \int_0^t a_2(s)ds + v_2^0 \\ x_2(t) &= F_{24}(a)(t) = \int_0^t v_2(s)ds + x_2^0. \end{aligned}$$

(d) **Network semantics:** The parameter tuple is

$$\mathbf{c} = (M_1, K_1, D_1, M_2, K_2, D_2, v_1^0, x_1^0, v_2^0, x_2^0),$$

the single input stream is again f , and the non-input stream tuple is (compare (4.2)):

$$\mathbf{u} = (a_1, v_1, x_1, a_2, v_2, x_2).$$

So we want a fixed point of the function

$$\Psi_{\mathbf{c},f}: \mathcal{C}[\mathbb{T}, \mathbb{R}]^6 \rightarrow \mathcal{C}[\mathbb{T}, \mathbb{R}]^6$$

where

$$\Psi_{\mathbf{c},f}(a_1, v_1, x_1, a_2, v_2, x_2) = (a'_1, v'_1, x'_1, a'_2, v'_2, x'_2)$$

with

$$a'_1(t) = \frac{1}{M_1}[f(t) - K_1(x_1(t) - x_2(t)) - D_1v_1(t)] \quad (4.12a)$$

$$v'_1(t) = \int_0^t a_1(s)ds + v_1^0 \quad (4.12b)$$

$$x'_1(t) = \int_0^t v_1(s)ds + x_1^0 \quad (4.12c)$$

$$a'_2(t) = \frac{1}{M_2}[K_1(x_1(t) - x_2(t)) - K_2x_2(t) - D_2v_2(t)] \quad (4.12d)$$

$$v'_2(t) = \int_0^t a_2(s)ds + v_2^0 \quad (4.12e)$$

$$x'_2(t) = \int_0^t v_2(s)ds + x_2^0. \quad (4.12f)$$

Again we must find under what conditions, $\Psi_{\mathbf{c},f}$ is contracting.

For changes $\delta a_1, \delta v_1, \dots$ in a_1, v_1, \dots , and corresponding changes $\delta a'_1, \dots$ in a'_1, \dots :

$$\Psi_{c,f}(a_1 + \delta a_1, v_1 + \delta v_1, \dots) = (a'_1 + \delta a'_1, v'_1 + \delta v'_1, \dots).$$

Note also from (4.12a, d) (taking $\delta f = 0$):

$$\begin{aligned} \delta a'_1 &= -\frac{1}{M_1}[K_1(\delta x_1 - \delta x_2) + D_1\delta v_1] \\ \delta a'_2 &= -\frac{1}{M_2}[K_1(\delta x_2 - \delta x_1) + K_2\delta x_2 + D_2\delta v_2] \end{aligned}$$

Hence for any $T \geq 0$ and $\tau > 0$, and again defining the pseudonorm $\|u\|$ as in (4.4), we have

$$\|\delta a'_1\| \leq \frac{K_1}{M_1}\|\delta x_1\| + \frac{K_1}{M_1}\|\delta x_2\| + \frac{D_1}{M_1}\|\delta v_1\| \quad (4.13a)$$

$$\|\delta a'_2\| \leq \frac{K_1}{M_2}\|\delta x_1\| + \left(\frac{K_1 + K_2}{M_2}\right)\|\delta x_2\| + \frac{D_2}{M_2}\|\delta v_2\| \quad (4.13b)$$

Also, from (4.12b, c, e, f):

$$\|\delta v'_1\| \leq \tau\|\delta a_1\| \quad (4.13c)$$

$$\|\delta x'_1\| \leq \tau\|\delta v_1\| \quad (4.13d)$$

$$\|\delta v'_2\| \leq \tau\|\delta a_2\| \quad (4.13e)$$

$$\|\delta x'_2\| \leq \tau\|\delta v_2\| \quad (4.13f)$$

Now assume (compare (4.6))

$$\begin{aligned} M_1 &> \max(2K_1, 2D_1) \\ M_2 &> \max(2K_1 + 2K_2, 2D_2) \end{aligned} \quad (4.14)$$

and put

$$\lambda =_{df} \max\left(\frac{K_1}{M_1} + \frac{K_1 + K_2}{M_2}, \frac{2D_1}{M_1}, \frac{2D_2}{M_2}\right). \quad (4.15)$$

Then by (4.14)

$$\lambda < 1. \quad (4.16)$$

Let

$$\tau = \min\left(\frac{D_1}{M_1}, \frac{D_2}{M_2}\right) \quad (4.17)$$

Then, defining the product pseudonorm as in (4.10), we have:

$$\begin{aligned}
& \|(\delta a'_1, \delta v'_1, \delta x'_1, \delta a'_2, \delta v'_2, \delta x'_2)\| \\
&= \|\delta a'_1\| + \|\delta v'_1\| + \|\delta x'_1\| + \|\delta a'_2\| + \|\delta v'_2\| + \|\delta x'_2\| \\
&\leq \left(\frac{K_1}{M_1} + \frac{K_1}{M_2}\right)\|\delta x_1\| + \left(\frac{D_1}{M_1} + \tau\right)\|\delta v_1\| + \tau\|\delta a_1\| \\
&\quad + \left(\frac{K_1}{M_1} + \frac{K_1 + K_2}{M_2}\right)\|\delta x_2\| + \left(\frac{D_2}{M_2} + \tau\right)\|\delta v_2\| + \tau\|\delta a_2\| \quad \text{by (4.13)} \\
&= \left(\frac{K_1}{M_1} + \frac{K_1}{M_2}\right)\|\delta x_1\| + \frac{2D_1}{M_1}\|\delta v_1\| + \frac{D_1}{M_1}\|\delta a_1\| \\
&\quad + \left(\frac{K_1}{M_1} + \frac{K_1 + K_2}{M_2}\right)\|\delta x_2\| + \frac{2D_2}{M_2}\|\delta v_2\| + \frac{D_2}{M_2}\|\delta a_2\| \quad \text{by (4.17)} \\
&\leq \lambda \|(\delta a_1, \delta v_1, \delta x_1, \delta a_2, \delta v_2, \delta x_2)\| \quad \text{by (4.15)}
\end{aligned}$$

which, by (4.16), proves $\mathbf{Contr}_{c,f}(\lambda)$.

Again, this gives a theorem (compare Theorem 3):

Theorem 4. *The network of Figure 8 is contracting, and hence satisfies \mathbf{NetDet} , for any input stream $f(t)$, provided*

$$M_1 > \max(2K_1, 2D_1) \quad \text{and} \quad M_2 > \max(2K_1 + 2K_2, 2D_2).$$

Corollary 4.2.1. *The system of Figure 8 has a well determined solution*

$$(a_1(t), v_1(t), x_1(t), a_2(t), v_2(t), x_2(t))$$

for the accelerations, velocities and displacements of the masses M_1 and M_2 as functions of time $t \geq 0$, given any input force $f(t)$ as a continuous function of time $t \geq 0$, and any initial conditions $(v_1^0, x_1^0, v_2^0, x_2^0)$ for the velocity and displacement of M_1 and M_2 , provided only that

$$M_1 > \max(2K_1, 2D_1) \quad \text{and} \quad M_2 > \max(2K_1 + 2K_2, 2D_2).$$

5 Computability of the solution

We want to show that the network function which solves the network specification (2.2) according to Theorem 3.2.2 is computable relative to the module functions for that network; in other words, the output streams are computable from the input streams, the parameters, and the module functions. Hence if all the module functions are computable, then so is the network function.

By “computable” here we mean: computable according to some concrete model of computation on $\mathcal{C}[\mathbb{T}, A]$. Computable stream transformations on $\mathcal{C}[\mathbb{T}, A]$ have been studied

using domains in [BSHT98]. An alternative treatment of concrete computation on the space $\mathcal{C}[X, Y]$ with the compact-open topology can be found in [Wei00], in the case that $X \subseteq \mathbb{R}^m$ and $Y = \mathbb{R}^n$. Here we will give a new model, inspired by the approximation of $\mathcal{C}[\mathbb{T}, A]$ by the metric spaces $\mathcal{C}[[0, k], A]$.

For reasons to be explained below, we will assume that A is separable.

5.1 Topological algebras of continuous streams

For our investigation of computation on $\mathcal{C}[\mathbb{T}, A]$, we must consider the *many-sorted topological algebra*

algebra	$\mathcal{C}[\mathbb{T}, A]$
carriers	$A, \mathbb{R}, \mathbb{T}, \mathcal{C}[\mathbb{T}, A], \mathbb{N}$
functions	$d_A: A^2 \rightarrow \mathbb{R},$ $\text{eval}: \mathcal{C}[\mathbb{T}, A] \times \mathbb{T} \rightarrow A$
end	

where $d_A: A^2 \rightarrow \mathbb{R}$ and $\text{eval}: \mathcal{C}[\mathbb{T}, A] \times \mathbb{T} \rightarrow A$ are, respectively, the distance function on A and the evaluation function

$$\text{eval}(u, t) = u(t).$$

We call the algebra $\mathcal{C}[\mathbb{T}, A]$ or just \mathcal{C} . It is a topological algebra, because each of the five carriers has an associated topology, as we have described (the usual one for \mathbb{R} , and the discrete one for \mathbb{N}), with respect to which the basic functions (d_A and eval) are continuous.

The carrier \mathbb{R} is needed for the metric operation on A .

The set of sorts of the signature Σ of \mathcal{C} is

$$\mathbf{Sort} = \mathbf{Sort}(\Sigma) = \{A, \mathbb{R}, \mathbb{T}, \mathcal{C}, \mathbb{N}\}$$

For ease of notation, we also refer to the five carriers of \mathcal{C} as C_s for $s \in \mathbf{Sort}$.

5.2 Enumerations of subfamilies of \mathcal{C}

The following is an extension of the concepts in [TZ04, §§6,7] on concrete computation on metric algebras to the case of the non-metric, topological algebra $\mathcal{C}[\mathbb{T}, A]$. We repeat some of the definitions there.

We will fix an *enumeration* of certain subsets of the carriers, *i.e.*, a family α of surjections

$$\alpha_s: \mathbb{N} \twoheadrightarrow X_s \subseteq C_s \quad (s \in \mathbf{Sort})$$

of \mathbb{N} with certain subsets X_s of C_s . The pair (X_s, α_s) is called an *enumerated subset* of C_s for $s \in \mathbf{Sort}$.

The enumerations are as follows. First, the mapping

$$\alpha_A: \mathbb{N} \twoheadrightarrow X \subseteq A$$

is an enumeration of some dense subset X of A . It is here that we need a separability assumption, failing which any enumerated subset of A could not be dense in A , thus trivialising the concrete model.

Assumption 5.2.1. *A is separable.*

This enumeration α_A (or rather its “computational closure” $\bar{\alpha}_A$, see below) must also satisfy a Σ -*effectivity* property, to be described below (5.4.4).

From Assumption 5.2.1 it follows that $\mathcal{C}[\mathbb{T}, A]$ is also separable [TZ07]. However we will need a stronger assumption than mere separability of $\mathcal{C}[\mathbb{T}, A]$, namely *effective local uniform continuity* of the countable dense subset of $\mathcal{C}[\mathbb{T}, A]$ (see Assumption 5.2.2 below).

The mapping

$$\alpha_{\mathbb{R}}: \mathbb{N} \rightarrow \mathbb{Q} \subset \mathbb{R}$$

is a standard enumeration of the rationals. (In case $A = \mathbb{R}$, α_A would be the same as $\alpha_{\mathbb{R}}$.) Similarly

$$\alpha_{\mathbb{T}}: \mathbb{N} \rightarrow \mathbb{Q}^+ \subset \mathbb{T}$$

is a standard enumeration of the non-negative rationals. The mapping

$$\alpha_{\mathbb{N}}: \mathbb{N} \rightarrow \mathbb{N}$$

is just the identity on \mathbb{N} . Finally, and most interestingly, the mapping

$$\alpha_{\mathcal{C}}: \mathbb{N} \rightarrow Z \subset \mathcal{C}[\mathbb{T}, A]$$

is a “standard” enumeration of some countable dense subset Z of $\mathcal{C}[\mathbb{T}, A]$, which must satisfy the Σ -*effectivity* property (5.4.4) referred to above, as well as the following

Assumption 5.2.2 (Effective locally uniform continuity for $(Z, \alpha_{\mathcal{C}})$). *There is a recursive function $\mu: \mathbb{N}^3 \rightarrow \mathbb{N}$ (an effective locally uniform modulus function) such that for all n, k, ℓ , writing $z_n = \alpha_{\mathcal{C}}(n)$:*

$$\forall t_1, t_2 \in [0, k]: |t_1 - t_2| < 2^{-\mu(n, k, \ell)} \Rightarrow d_A(z_n(t_1), z_n(t_2)) < 2^{-\ell},$$

or, more simply but equivalently: There is a recursive function $\mu': \mathbb{N}^2 \rightarrow \mathbb{N}$ such that for all n, k , writing $z_n = \alpha_{\mathcal{C}}(n)$:

$$\forall t_1, t_2 \in [0, k]: |t_1 - t_2| < 2^{-\mu'(n, k)} \Rightarrow d_A(z_n(t_1), z_n(t_2)) < 2^{-k}.$$

5.3 Computational closure

For our model of *concrete computation* on $\mathcal{C}[\mathbb{T}, A]$, we are interested in the *computational closures* $\mathcal{C}_{\alpha_s}(X_s)$ of the enumerated subsets (X_s, α_s) of the spaces C_s ($s \in \mathbf{Sort}$), with enumerations

$$\bar{\alpha}_s: \Omega_{\bar{\alpha}_s} \rightarrow \mathcal{C}_{\alpha_s}(X_s)$$

so that

$$X_s \subseteq \mathcal{C}_{\alpha_s}(X_s) \subseteq C_s \quad (s \in \mathbf{Sort})$$

as we now describe.

First, for the complete metric space A , let us define the set $\mathcal{C}_{\alpha_A}(X)$ of α -computable elements of A , to be the limits in A of effectively convergent Cauchy sequences of elements of the enumerated subset X , with the corresponding enumeration

$$\bar{\alpha}_A : \Omega_{\bar{\alpha}_A} \rightarrow \mathcal{C}_{\alpha_A}(X).$$

Details of the construction of $\mathcal{C}_{\alpha_A}(X)$ and $\bar{\alpha}_A$ can be found in [TZ04]. We omit them, since below, for the computational closure of $Z \in \mathcal{C}[\mathbb{T}, A]$, we describe a model of concrete computability for a more general situation — the non-metric topological space $\mathcal{C}[\mathbb{T}, A]$.

The computational closures $\mathcal{C}_{\alpha_{\mathbb{R}}}(\mathbb{Q})$ and $\mathcal{C}_{\alpha_{\mathbb{T}}}(\mathbb{Q}^+)$ in \mathbb{R} and \mathbb{T} respectively are defined in the same way.

The computational closure of \mathbb{N} is, trivially, \mathbb{N} , with (again) the identity enumeration.

Finally, for the space $\mathcal{C}[\mathbb{T}, A]$ with its enumerated subset (Z, α_Z) (where we henceforth usually drop the subscripts of α and $\bar{\alpha}$), let

$$\mathcal{C}_{\alpha}(\mathbb{T}, A) =_{df} \mathcal{C}_{\alpha}(Z) \subset \mathcal{C}[\mathbb{T}, A]$$

be the set of all limits in $\mathcal{C}[\mathbb{T}, A]$ of α -effectively locally uniform Cauchy sequences of elements of Z — such limits always existing by the completeness of $\mathcal{C}[\mathbb{T}, A]$ (Lemma 3.1.4) — and let $\Omega_{\bar{\alpha}} \subset \mathbb{N}$ be the set of codes for $\mathcal{C}_{\alpha}(\mathbb{T}, A)$. More precisely, $\Omega_{\bar{\alpha}}$ consists of pairs of numbers $c = \langle e, m \rangle$ where

(i) e is an index for a total recursive function defining a sequence

$$z_0, z_1, z_2, \dots \tag{5.1}$$

of elements of Z , where $z_n = \alpha(\{e\}(n))$, and

(ii) m is an index for a modulus of local uniform convergence for this sequence; i.e., for all k :

$$\forall n, p \geq \{m\}(k), \forall t \in [0, k] : d_A(z_n(t), z_p(t)) \leq 2^{-k}$$

or, equivalently:

$$\forall n, p \geq \{m\}(k), d_k(z_n \upharpoonright_k, z_p \upharpoonright_k) \leq 2^{-k}. \tag{5.2}$$

For any such code c , $\bar{\alpha}(c)$ is defined as the limit in $\mathcal{C}[\mathbb{T}, A]$ of the Cauchy sequence (5.1), and $\mathcal{C}_{\alpha}(\mathbb{T}, A)$ is the range of $\bar{\alpha}$:

$$\begin{array}{ccc} Z & \subset & \mathcal{C}_{\alpha}(\mathbb{T}, A) \subset \mathcal{C}[\mathbb{T}, A] \\ \uparrow \alpha & & \uparrow \bar{\alpha} \\ \mathbb{N} & & \Omega_{\bar{\alpha}} \end{array}$$

The effective locally uniform continuity property (Assumption 5.2.2) “lifts” from (Z, α_C) to $(\mathcal{C}_\alpha(\mathbb{T}, A), \bar{\alpha}_C)$:

Lemma 5.3.1 (Effective locally uniform continuity for $\mathcal{C}_\alpha(\mathbb{T}, A)$). *If (Z, α_C) satisfies effective locally uniform continuity, then so does $(\mathcal{C}_\alpha(\mathbb{T}, A), \bar{\alpha}_C)$*

Proof: An effective locally uniform continuity modulus function for $(\mathcal{C}_\alpha(\mathbb{T}, A), \bar{\alpha}_C)$ can be constructed from the one for (Z, α_C) (see Assumption 5.2.2) by essentially constructivising the classical proof [Rud76] of the theorem that a limit of a uniform Cauchy sequence of uniformly continuous functions is uniformly continuous. \square

Note that in the case of $(\mathcal{C}_\alpha(\mathbb{T}, A), \bar{\alpha}_C)$, the effectively locally uniform modulus function $\bar{\mu}$ is partial, defined only on inputs (n, \dots) for which $n \in \Omega_{\bar{\alpha}}$.

5.4 Concrete computation on $\mathcal{C}[\mathbb{T}, A]$

For a tuple of sorts $\sigma = (s_1, \dots, s_m)$, we have the *product space*

$$C^\sigma =_{df} C_{s_1} \times \dots \times C_{s_m},$$

and *product domain*

$$\Omega_{\bar{\alpha}}^\sigma =_{df} \Omega_{\bar{\alpha}_{s_1}} \times \dots \times \Omega_{\bar{\alpha}_{s_m}} \subseteq \mathbb{N}^m$$

and define the *product enumeration*

$$\bar{\alpha}^\sigma = (\alpha_{s_1}, \dots, \alpha_{s_m}): \Omega_{\bar{\alpha}}^\sigma \rightarrow C^\sigma$$

in the obvious way.

We are interested in (partial) functions on $\mathcal{C}[\mathbb{T}, A]$ of type

$$f: C^\sigma \rightarrow C_s.$$

Definition 5.4.1 (Tracking function). Let $f: C^\sigma \rightarrow C_s$, where $\sigma = (s_1, \dots, s_m)$. A function $\varphi: \Omega_{\bar{\alpha}}^\sigma \rightarrow \Omega_{\bar{\alpha}_s}$ is an $\bar{\alpha}$ -*tracking function* for f if the following diagram commutes:

$$\begin{array}{ccc} C^\sigma & \xrightarrow{f} & C_s \\ \bar{\alpha}^\sigma \uparrow & & \uparrow \bar{\alpha}_s \\ \Omega_{\bar{\alpha}}^\sigma & \xrightarrow{\varphi} & \Omega_{\bar{\alpha}_s} \end{array}$$

in the sense that for all $k \in \Omega_{\bar{\alpha}}^\sigma$,

$$f(\bar{\alpha}^\sigma(k)) \downarrow \implies \varphi(k) \downarrow \wedge \varphi(k) \in \Omega_{\bar{\alpha}_s} \wedge f(\bar{\alpha}^\sigma(k)) = \bar{\alpha}_s(\varphi(k)).$$

Definition 5.4.2 (Concrete computability on $\mathcal{C}[\mathbb{T}, A]$). Suppose f, g_1, \dots, g_k are functions on $\mathcal{C}[\mathbb{T}, A]$ with $\bar{\alpha}$ -tracking functions $\varphi, \psi_1, \dots, \psi_k$ respectively. Then f is $\bar{\alpha}$ -*computable in* (or *relative to*) g_1, \dots, g_k iff φ is partially recursive in ψ_1, \dots, ψ_k .

Remarks 5.4.3.

(a) (*Locally fast Cauchy sequences.*) We may assume, when convenient, that the modulus of convergence for a given code is the *identity*, i.e., replace (5.2) by the simpler

$$\forall m, l \geq n : d_k(z_m \upharpoonright_n, z_l \upharpoonright_n) < 2^{-n},$$

or equivalently,

$$\forall m > n : d_k(z_m \upharpoonright_n, z_n \upharpoonright_n) < 2^{-n}, \quad (5.3)$$

because any code $c = \langle e, m \rangle$ satisfying (5.2) can be effectively replaced by a code for the same element of $\mathcal{C}_\alpha(\mathbb{T}, \mathbb{R})$ satisfying (5.3), namely $c' = \langle e', m_1 \rangle$, where m_1 is a standard code for the identity function on \mathbb{N} , and $\{e'\}(n) = \{e\}(\{m\}(n)) = z_{\{m\}(n)}$. In the case of a code $c = \langle e, m_1 \rangle$ satisfying (5.3) (with $\alpha(\{e\}(n)) = z_n$), the sequence (5.1) is called a *locally fast Cauchy sequence*. We may then, for simplicity, call e itself the “code”, and the argument of $\bar{\alpha}$. So we can shift between “ c -codes” and “ e -codes” as convenient.

(b) (*Computational closure of $\mathcal{C}_\alpha(\mathbb{T}, A)$.*) The subspace $\mathcal{C}_\alpha(\mathbb{T}, A)$ is computationally closed in $\mathcal{C}[\mathbb{T}, A]$, in the sense that the limit of a $\bar{\alpha}$ -effectively locally uniformly Cauchy sequence of elements of $\mathcal{C}_\alpha(\mathbb{T}, A)$ is again in $\mathcal{C}_\alpha(\mathbb{T}, A)$, i.e., $\mathcal{C}_{\bar{\alpha}}(\mathcal{C}_\alpha(\mathbb{T}, A)) = \mathcal{C}_\alpha(\mathbb{T}, A)$.

There is one more assumption needed on the choice of the enumerations α (specifically α_A and α_C) from which $\bar{\alpha}$ was constructed:

Assumption 5.4.4 (Σ -effectivity of $\bar{\alpha}$). *The basic functions of the algebra $\mathcal{C}[\mathbb{T}, A]$, namely d_A and eval , are $\bar{\alpha}$ -computable.*

This is used in the proof of the following corollary, which in turn is used in the proof of Theorem 5.

Corollary 5.4.5. *The function*

$$f : \mathbb{N} \times \mathcal{C}[\mathbb{T}, A]^m \times \mathcal{C}[\mathbb{T}, A]^m \rightarrow \mathbb{R}$$

defined by

$$f(k, \mathbf{u}, \mathbf{v}) = d_k(\mathbf{u}, \mathbf{v})$$

is $\bar{\alpha}$ -computable.

Proof: It is required to find, $\bar{\alpha}$ -effectively and uniformly in k ,

$$\sup_{0 \leq t \leq k} d_A(\mathbf{u}(t), \mathbf{v}(t)).$$

For this we can follow the technique of [PER89] (Chapter 0, Theorem 7) using the effective locally uniform continuity of \mathbf{u} and \mathbf{v} . By Assumption 5.4.4 the function d_A in the above expression is $\bar{\alpha}$ -computable, as is the function $\text{eval}^m(\mathbf{u}, t) =_{df} \mathbf{u}(t)$. \square

Example 5.4.6 (Concrete computation on $\mathcal{C}[\mathbb{T}, \mathbb{R}]$). Consider, in particular, the case that the metric space A is \mathbb{R} . As stated above, for α_A we would take the same as $\alpha_{\mathbb{R}}$, *i.e.*, a standard enumeration of the rationals.

As an example of a countable and locally uniformly dense subset of $\mathcal{C}[\mathbb{T}, \mathbb{R}]$, take $Z = \text{ZZ}$, the set of all *continuous rational “zigzag functions”* from \mathbb{T} to \mathbb{R} with *finite support*, a typical example of which is shown in Figure 10, where we require that the starting and turning points (p_1, \dots, p_7 in the figure) have rational coordinates, and which are zero from some point on (p_7 in the figure).

It is clear that the set ZZ , under any reasonable enumeration α_C , satisfies the effective locally uniform continuity assumption (5.2.2). Also, the enumeration $\bar{\alpha}$ derived from α is clearly Σ -effective (Assumption 5.4.4).

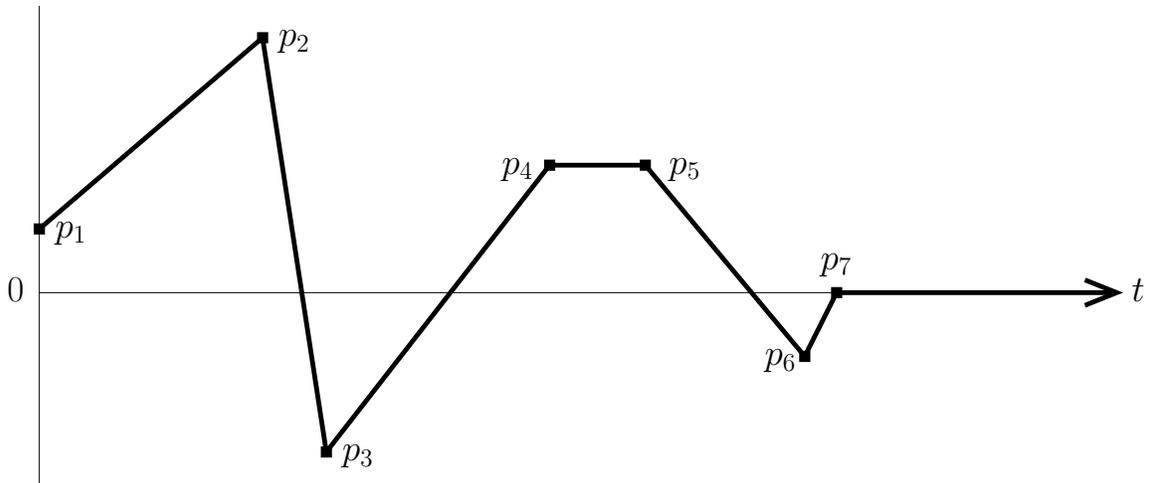


FIGURE 10: Zigzag function (points p_1, \dots, p_7 are rational)

Note that we could have used, as our starting point, the set of polynomial functions of t with rational coefficients. This would produce the same set $\mathcal{C}_\alpha(\mathbb{T}, \mathbb{R})$ of computable elements of $\mathcal{C}[\mathbb{T}, \mathbb{R}]$.

5.5 Relative concrete computability of functions defined by analog networks

Given a network N as in §2.3, we want to show that the network function Φ^N is $\bar{\alpha}$ -computable relative to the module functions, provided it is contracting at the parameter and stream inputs.

For this we need a constructive concept of contraction, namely that a contraction modulus $\lambda < 1$ can be found *effectively* in the parameters \mathbf{c} and stream inputs \mathbf{x} .

Definition 5.5.1 (Effectively contracting network). Given a subset $U \subseteq A^r \times \mathcal{C}[\mathbb{T}, A]^p$, the network N is $(\bar{\alpha})$ -*effectively contracting* on U if a contraction modulus $\lambda_{\mathbf{c}, \mathbf{x}}$ can be found $\bar{\alpha}$ -effectively in $(\mathbf{c}, \mathbf{x}) \in U$.

This means that there is a recursive partial function $\varphi: \mathbb{N}^r \times \mathbb{N}^p \rightarrow \mathbb{N}$ which $\bar{\alpha}$ -tracks $\lambda_{\mathbf{c}, \mathbf{x}}$ as a function of (\mathbf{c}, \mathbf{x}) restricted to U .

Note that this is certainly the case with the two case studies in Section 4. In Case Study 1, for example, a value for λ can be found effectively in the parameters M, K, D (and independent of the input stream f), by equation (4.7), in the region

$$U =_{df} \{ (M, K, D) \in \mathbb{R}^3 \mid M > \max(K, D) \}.$$

Similarly for Case Study 2.

We want to prove the following.

Theorem 5. *Suppose the network N satisfies the condition **Caus**, the enumerated sets (X, α_A) and (Z, α_C) are dense in A and $\mathcal{C}[\mathbb{T}, A]$ respectively, (Z, α_C) satisfies effective local uniform continuity, and $\bar{\alpha}$ is Σ -effective. Suppose also N is $\bar{\alpha}$ -effectively contracting on $U \subseteq A^r \times \mathcal{C}[\mathbb{T}, A]^p$. Then the network function*

$$\Phi^N: A^r \times \mathcal{C}[\mathbb{T}, A]^p \rightarrow \mathcal{C}[\mathbb{T}, A]^q$$

is defined (at least) on U , and is $\bar{\alpha}$ -computable relative to the module functions of N . Hence if the module functions are $\bar{\alpha}$ -computable, then so is Φ^N .

From now on, by “computable” we mean $\bar{\alpha}$ -computable.

Consider, then, a network N satisfying **Caus**, and enumerations $\bar{\alpha}$ of $\mathcal{C}[\mathbb{T}, A]$ satisfying effective local uniform continuity and Σ -effectivity. Recall the definitions and notation in §3.2.

Lemma 5.5.2. *The composition of two computable functions is computable.*

Lemma 5.5.3. *The function*

$$\Psi^N: A^r \times \mathcal{C}[\mathbb{T}, A]^p \times \mathcal{C}[\mathbb{T}, A]^{m-p} \rightarrow \mathcal{C}[\mathbb{T}, A]^{m-p}$$

defined by

$$\Psi^N(\mathbf{c}, \mathbf{x}, \mathbf{u}) = \Psi_{\mathbf{c}, \mathbf{x}}^N(\mathbf{u}) = (F_{p+1}(\mathbf{u}_1, \mathbf{c}_1), \dots, F_m(\mathbf{u}_m, \mathbf{c}_m))$$

(cf. (3.10), (3.11)) is computable relative to F_1, \dots, F_m .

We need an effective version of the notion of locally uniform Cauchy sequence (Definition 3.1.1).

Definition 5.5.4 (Effectively locally uniform Cauchy sequence). A sequence $(\mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_2, \dots)$ of elements of $\mathcal{C}[\mathbb{T}, A]^p$ is effectively locally uniformly Cauchy if there is a recursive function $\nu: \mathbb{N} \rightarrow \mathbb{N}$ such that for all k and all $m, n \geq \nu(k)$, $\mathbf{d}_k(\mathbf{u}_m, \mathbf{u}_n) \leq 2^{-k}$.

Lemma 5.5.5. *Let*

$$f: A^r \times \mathcal{C}[\mathbb{T}, A]^m \times \mathbb{N} \rightarrow \mathcal{C}[\mathbb{T}, A]$$

be a function such that for all $(\mathbf{c}, \mathbf{x}) \in U \subseteq A^r \times \mathcal{C}[\mathbb{T}, A]^m$ and all n , $f(\mathbf{c}, \mathbf{x}, n) \downarrow$, and for all $(\mathbf{c}, \mathbf{x}) \in U$ the sequence

$$f(\mathbf{c}, \mathbf{x}, 0), f(\mathbf{c}, \mathbf{x}, 1), f(\mathbf{c}, \mathbf{x}, 2), \dots$$

of elements of $\mathcal{C}[\mathbb{T}, A]$ is effectively locally uniformly Cauchy. Define

$$g: A^r \times \mathcal{C}[\mathbb{T}, A]^m \rightarrow \mathcal{C}[\mathbb{T}, A]$$

by

$$g(\mathbf{c}, \mathbf{x}) \simeq \lim_n f(\mathbf{c}, \mathbf{x}, n).$$

Then g is defined (at least) on U , and if f is computable, then so is g .

Proof: Essentially, one takes a “diagonal sequence” of approximations of the sequence $f(\mathbf{c}, \mathbf{x}, n)$ for $n = 0, 1, 2, \dots$ (as in the proof of Remark 5.4.3(b)). \square

Recall now that the network function

$$\Phi^N: A^r \times \mathcal{C}[\mathbb{T}, A]^p \rightarrow \mathcal{C}[\mathbb{T}, A]^q,$$

with

$$\Phi^N(\mathbf{c}, \mathbf{x}) \downarrow \mathbf{y} \quad (\text{say}),$$

assuming $(\mathbf{c}, \mathbf{x}) \in U$, is obtained from the fixed point \mathbf{u} of the function

$$\Psi_{\mathbf{c}, \mathbf{x}}^N: \mathcal{C}[\mathbb{T}, A]^{m-p} \rightarrow \mathcal{C}[\mathbb{T}, A]^{m-p},$$

i.e.,

$$\Psi_{\mathbf{c}, \mathbf{x}}^N(\mathbf{u}) = \mathbf{u}$$

since for a given input $(\mathbf{c}, \mathbf{x}) \in U$, the output \mathbf{y} of Φ^N is just a sub-tuple of this fixed point \mathbf{u} .

So it is sufficient to show that the function from $(\mathbf{c}, \mathbf{x}) \in U$ to this \mathbf{u} is computable (relative to the module functions) — or, as we will express it, \mathbf{u} is computable in (\mathbf{c}, \mathbf{x}) (relative to the module functions).

Now consider the sequence of stream tuples \mathbf{u}_n , defined in the proof of Theorem 1, taking $f = \Psi_{\mathbf{c}, \mathbf{x}}^N$, for some $(\mathbf{c}, \mathbf{x}) \in U$.

(1) The streams \mathbf{u}_n are computable in (\mathbf{c}, \mathbf{x}) , for all n . This is shown by induction on n :

Basis. For $n = 0$, this is clear: take for \mathbf{u}_0 any stream with a computable constant value.

Induction step. Suppose \mathbf{u}_n is computable in (\mathbf{c}, \mathbf{x}) (relative to the module functions). Then $\mathbf{u}_{n+1} = \Psi_{\mathbf{c}, \mathbf{x}}^N(\mathbf{u}_n)$ is also computable in (\mathbf{c}, \mathbf{x}) (relative to the module functions) by Lemmas 5.5.3 and 5.5.2.

(2) Further, (\mathbf{u}_n) is an *effectively* locally uniform Cauchy sequence. This is because, by assumption, a value for the contraction modulus λ can be found *effectively* in $(\mathbf{c}, \mathbf{x}) \in U$.

Hence a value for N in the inequality (3.8), as a witness to the local uniform convergence of this sequence, can be found *effectively in* (\mathbf{c}, \mathbf{x}) (relative to the module functions) by Corollary 5.4.5. Hence, by Lemma 5.5.5, the limit \mathbf{u} of this sequence, which is the fixed point of the function $\Psi_{\mathbf{c}, \mathbf{x}}^N$ as desired, is also computable in (\mathbf{c}, \mathbf{x}) (relative to the module functions).

This completes the proof of Theorem 5.

5.6 Concrete computability of module functions

We show that various standard module functions on $\mathcal{C}[\mathbb{T}, \mathbb{R}]$ are $\bar{\alpha}$ -computable.

- *The identity function*
- *Addition*
- *Multiplication by a constant*

These are all obvious.

- *Integration.* This is the interesting case. Suppose

$$F(u, c) = v, \quad \text{where} \quad v(t) = \left(\int_0^t u \right) + c.$$

By the effective locally uniform continuity assumption (5.2.2, Version 2) for $\alpha_{\mathcal{C}}$, applied to $\bar{\alpha}_{\mathcal{C}}$ by Lemma 5.3.1, let μ be the effective locally uniform continuity modulus function for $\mathcal{C}[\mathbb{T}, \mathbb{R}]$. Define

$$g: \mathcal{C}[\mathbb{T}, \mathbb{R}] \times \mathbb{N} \rightarrow \mathcal{C}[\mathbb{T}, \mathbb{R}]$$

by

$$g(u, n) = v_n,$$

where $v_n(t)$ is the Riemann sum for u from 0 to t , using rectangles of width $\leq 2^{-\mu(e, 2n)}$, where e is a code for u . Then

$$d_n(v_n, v) \leq 2^{-2n} \cdot n < 2^{-n},$$

and so (v_n) is a locally fast Cauchy sequence, with limit v . Since g is easily seen to be computable, so is F , by Lemma 5.5.5.

It follows that all the module functions in the case studies in Section 4 are computable. Combining this with Theorem 5, we conclude that the functions which solve the network equations in these two case studies are computable.

6 Concluding Remarks

The contemporary literature is focussed on mathematically modelling analog computation and characterising its computational power, commonly using schemes to define classes of functions on the real numbers. The base functions of the schemes are normally the traditional functions of analog computation (adders, integrators etc.).

Our network models, involving arbitrary processing units on arbitrary data in continuous time, are new. They emphasise the fundamental role of a physical technology on which analog computation is based.

6.1 What is analog computation?

A comprehensive model of analog computation should cover these six aspects:

1. Identifying the motivating problem P .
2. Specifying the problem P by a system E of differential equations.
3. Designing a network N from P to solve E .
4. Calculating conditions on the data and parameters of P to ensure good experimental behaviour of N .
5. Constructing an analog machine M from N using a particular technology.
6. Using the machine M for measurements/experimental procedures.

Our network model concentrates on aspects 3 and 4, though in our case studies in Section 4 we consider aspects 1 and 2, and comment upon the implementation aspects 5 and 6.

6.2 Problems about networks, related to the present research

Several questions and problems concerning analog networks are left open:

1. The modules of our theory are essentially “black boxes”. For the module functions, it turned out (surprisingly?) that we did not need the assumption of *time invariance* at all. This assumption, as well as *causality* (which we did need) are both satisfied by the standard module functions listed in §5.6, and are common in dynamical system theory [Hay89, OW97]. What is the significance of such assumptions — or their absence — in the present setting?
2. Nor did we need the assumption of *continuity* of the modules for the existence of the network function (part (a) of Theorem 2), only for its continuity (part(b)). Again, this assumption is satisfied by the standard module functions listed in §5.6. Note also that the input and output streams are continuous. The interest of all this is related to Hadamard’s principle (as already noted in Remark 3.3.2). What happens if we weaken our continuity assumption for modules — or for streams — by postulating, for example, *piecewise continuous streams*? This is a common natural phenomenon: consider phase changes in thermodynamics.
3. What is the consequence of allowing
 - (a) module functions that are *partial* or *many-valued*?
 - (b) partial streams?
4. Find reasonable conditions, other than the contraction property, that guarantee “good behaviour” of these networks.
5. Characterise the networks that produce all (and only) concretely computable functions on $\mathcal{C}[\mathbb{T}, A]$.

To the above technical questions can be added one of a more conceptual nature:

6. How can we be certain that our constructed solution, as the fixed point of a certain function, indeed represents the behaviour of the network N — or, for that matter, that N faithfully represents the original physical system P ? Eugene Wigner’s celebrated paper [Wig60] may be relevant here, at least for the first question.

6.3 General problems concerning analog computation

Widening our considerations from networks to other paradigms of analog computation, we have the following two general problems, complementing the two questions about analog technology given in the Introduction:

1. **Classification of analog models:** Classify the disparate models of analog computation, by establishing equivalences or inequivalences between them.

This would seem to require a substantial research programme. In addition to schemes [Moo96] and systems of differential equations [Sha41, PE74] there are models with quite different motivations. For example, a generalisation of the theory of finite state automata to continuous time has been proposed by Trakhtenbrot, Rabinovich and others [Tra66, RT98, Rab03], which features data signals modelled by piecewise constant functions. In addition, there are models of continuous neural nets that are relevant for any general theory of analog computation [Cul91]. Finally, control systems provide a wealth of theories and examples that must be considered in a comprehensive classification programme [Son90].

2. **Technology integration:** How can analog and digital models be integrated, either in hybrid systems, or in analog implementations of digital systems, or in digital implementations of analog systems?

We conclude with a question of a more speculative or philosophical nature, possibly related to question 6 of the previous subsection.

3. Given that scientific measurements produce rational numbers, how is it that such experimental procedures, applied to analog machines, can calculate functions on the real continuum?

Clearly there is a great deal to theorise about.

References

- [BCGH06] O. Bournez, M.L. Campagnolo, D.S. Graça, and E. Hainry. The general purpose analog computer and computable analysis are two equivalent paradigms of analog computation. In J.-Y. Cai, S.B. Cooper, and A. Li, editors, *Theory and Applications of Models of Computation: Third International Conference, TAMC 2006, Beijing, China, May 15-20, 2006. Proceedings*, volume 3959 of *Lecture Notes in Computer Science*, pages 631–643. Springer-Verlag, 2006.
- [BCGH07] O. Bournez, M.L. Campagnolo, D.S. Graça, and E. Hainry. Polynomial differential equations compute all real computable functions. *Journal of Complexity*, 23:317–335, 2007.
- [Bee85] M. Beeson. *Foundations of Constructive Mathematics*. Springer-Verlag, 1985.

- [BSHT98] J. Blanck, V. Stoltenberg-Hansen, and J.V. Tucker. Domain representation of streams. In B. Möller and J.V. Tucker, editors, *Prospects for hardware foundations*, volume 1546 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.
- [Bus31] V. Bush. The differential analyzer. A new machine for solving differential equations. *Journal of the Franklin Institute*, 212:447–488, 1931.
- [CH53] R. Courant and D. Hilbert. *Methods of Mathematical Physics, Vol. II*. Interscience, 1953. Translated and revised from the German [1937].
- [CMC02] M. Campagnolo, C. Moore, and J.F. Costa. An analog characterization of the Grzegorzcyk hierarchy. *Journal of Complexity*, 18:977–1000, 2002.
- [Cul91] P. Cull. Dynamics of neural nets. *Trends in Biological Cybernetics*, 1:331–349, 1991.
- [Eng89] R. Engelking. *General Topology*. Heldermann Verlag, 1989.
- [GC03] D.S. Graça and J.F. Costa. Analog computers and recursive functions over the reals. *Journal of Complexity*, 19:644–664, 2003.
- [GCB05] D.S. Graça, M.L. Campagnolo, and J. Buescu. Robust simulations of Turing machines with analytic maps and flows. In S.B. Cooper, B. Löwe, and L. Torenvliet, editors, *New Computational Paradigms: Computability in Europe, CiE 2005, Amsterdam, June 2005*, volume 3526 of *Lecture Notes in Computer Science*, pages 169–179. Springer-Verlag, 2005.
- [Gri99] E. Griffor, editor. *Handbook of Computability Theory*. North Holland, 1999.
- [Had52] J. Hadamard. *Lectures on Cauchy’s Problem in Linear Partial Differential Equations*. Dover, 1952. Translated from the French [1922].
- [Had64] J. Hadamard. *La Théorie des Équations aux Dérivées Partielles*. Éditions Scientifiques, 1964.
- [Har50] D. Hartree. *Calculating machines and instruments*. Cambridge University Press, 1950.
- [Hay89] S. Haykin. *An introduction to analog and digital communications*. John Wiley & Sons, 1989.
- [Hol96] P.A. Holst. Svein Rosseland and the Oslo Analyzer. *IEEE Annals in the History of Computing*, 18:16–26, 1996.
- [HTT88] K.M. Hobbey, B.C. Thompson, and J.V. Tucker. Specification and verification of synchronous concurrent algorithms: a case study of a convoluted algorithm. In G. Milne, editor, *The Fusion of Hardware Design and Verification (Proceedings of IFIP Working Group 10.2 Working Conference)*, pages 347–374. North Holland, 1988.
- [Hyn70] D.E. Hyndman. *Analog and Hybrid Computing*. Pergamon Press, 1970.
- [Joh96] M. Johansson. Early analog computers in Sweden — with examples from Chalmers University of Technology and the Swedish aerospace industry. *IEEE Annals in the History of Computing*, 18:27–33, 1996.
- [Kle52] S.C. Kleene. *Introduction to Metamathematics*. North Holland, 1952.

- [LR87] L. Lipshitz and L.A. Rubel. A differentially algebraic replacement theorem. *Proceedings of the American Mathematical Society*, 99(2):367–372, 1987.
- [MC04] J. Mycka and J.F. Costa. Real recursive functions and their hierarchy. *Journal of Complexity*, 20:835–857, 2004.
- [Moo96] C. Moore. Recursion theory on the reals and continuous time computation. *Theoretical Computer Science*, 162:23–44, 1996.
- [Myr95] W.C. Myrvold. Computability in quantum mechanics. In W. DePauli-Schimanovich, E. Köhler, and F. Stadler, editors, *The Foundational Debate: Complexity and Constructivity in Mathematics and Physics*, pages 33–46. Kluwer, 1995.
- [OW97] A.V. Oppenheim and A.S. Willsky. *Signals and Systems (2nd ed.)*. Prentice Hall, 1997.
- [PE74] M.B. Pour-El. Abstract computability and its relation to the general-purpose analog computer. *Journal of the American Mathematical Society*, 199:1–28, 1974.
- [PER89] M.B. Pour-El and J.I. Richards. *Computability in Analysis and Physics*. Springer-Verlag, 1989.
- [Puc96] S. Puchta. On the role of mathematics and mathematical knowledge in the invention of Vannevar Bush’s early analog computers. *IEEE Annals in the History of Computing*, 18:49–56, 1996.
- [Rab03] A. Rabinovich. Automata over continuous time. *Theor. Computer Science*, 300:331–363, 2003.
- [Roy63] H.L. Royden. *Real Analysis*. Macmillan, 1963.
- [RT98] A. Rabinovich and B.A. Trakhtenbrot. From finite automata toward hybrid systems. In L. Brim, J. Gruska, and J. Zlatuska, editors, *Proceedings of the 23rd International Symposium on Mathematical Foundations of Computer Science (MFCS’98), Brno, Czech Republic, August 1998*, volume 1450 of *Lecture Notes in Computer Science*, pages 411–422. Springer-Verlag, 1998.
- [Rud76] W. Rudin. *Principles of Mathematical Analysis (3rd ed.)*. McGraw-Hill, 1976.
- [Sha41] C Shannon. Mathematical theory of the differential analyser. *Journal of Mathematics and Physics*, 20:337–354, 1941.
- [SHT99] V. Stoltenberg-Hansen and J.V. Tucker. Concrete models of computation for topological algebras. *Theoretical Computer Science*, 219:347–378, 1999.
- [Sma96] J.S. Small. General-purpose electronic analog computing: 1945–1965. *IEEE Annals in the History of Computing*, 18:8–18, 1996.
- [Son90] E.D. Sontag. *Mathematical Control Theory: Deterministic Finite Dimensional Systems*. Springer-Verlag, 1990.
- [Tra66] B.A. Trakhtenbrot. Finite automata and the logic of one-place predicates. *AMS Translations*, 59:23–55, 1966. Translated from the Russian [1961].
- [TT80] W. Thompson and P.G. Tait. *Treatise on Natural Philosophy (2nd ed.) Part I*. Cambridge University Press, 1880.

- [TT91] B.C. Thompson and J.V. Tucker. Algebraic specification of synchronous concurrent algorithms and architectures (Revised). Research Report 9-91, Department of Computer Science, Swansea University, Swansea, Wales, 1991.
- [TZ94] J.V. Tucker and J.I. Zucker. Computable functions on stream algebras. In H. Schwichtenberg, editor, *Proof and Computation: NATO Advanced Study Institute International Summer School at Marktoberdorf, 1993*, pages 341–382. Springer-Verlag, 1994.
- [TZ04] J.V. Tucker and J.I. Zucker. Abstract versus concrete computation on metric partial algebras. *ACM Transactions on Computational Logic*, 5:611–668, 2004.
- [TZ05] J.V. Tucker and J.I. Zucker. Computable total functions, algebraic specifications and dynamical systems. *Journal of Logic and Algebraic Programming*, 62:71–108, 2005.
- [TZ06] J.V. Tucker and J.I. Zucker. Abstract versus concrete computability: The case of countable algebras. In V. Stoltenberg-Hansen and J. Väänänen, editors, *Logic Colloquium '03, Proceedings of the Annual European Summer Meeting of the Association for Symbolic Logic, held in Helsinki, Finland, August 14–20, 2003*, volume 24 of *Lecture Notes in Logic*, pages 377–408. Association for Symbolic Logic, 2006.
- [TZ07] J.V. Tucker and J.I. Zucker. Computation on algebras of continuous functions. In preparation, 2007.
- [Wei00] K. Weihrauch. *Computable Analysis: An Introduction*. Springer-Verlag, 2000.
- [Wig60] E. Wigner. The unreasonable effectiveness of mathematics in the natural sciences. *Communications in Pure Mathematics*, 13:1–14, 1960.