Analog Networks on Function Data Streams

Diogo Poças & Jeffery Zucker
Department of Mathematics and Statistics
McMaster University, Hamilton, Ontario, Canada

November 21, 2016

Abstract

Most of the physical processes arising in nature are modeled by differential equations, either ordinary (example: the spring/mass/damper system) or partial (example: heat diffusion). From the point of view of analog computability, the existence of an effective way to obtain solutions (either exact or approximate) of these systems is essential.

A pioneering model of analog computation is the General Purpose Analog Computer (GPAC), introduced by Shannon [Sha41] as a model of the Differential Analyzer and improved by Pour-El [PE74], Lipshitz and Rubel [LR87], Costa and Graça [GC03] and others. The GPAC is capable of manipulating real-valued data streams. Its power is known to be characterized by the class of differentially algebraic functions, which includes the solutions of initial value problems for ordinary differential equations.

We address one of the limitations of this model, which is its fundamental inability to reason about functions of more than one independent variable (the 'time' variable), as noted by Rubel [Rub93]. In particular, the Shannon GPAC cannot be used to specify solutions of partial differential equations. We extend the class of data types using networks with channels which carry information on a general complete metric space X; here we take $X = C(\mathbb{R})$, the class of continuous functions of one real (spatial) variable.

We consider the original modules in Shannon's construction (constants, adders, multipliers, integrators) and we add a *differential* module which has one input and one output. For input u, it outputs the spatial derivative $v(t) = \partial_x u(t)$.

We then define an X-GPAC to be a network built with X-stream channels and the above-mentioned modules. This leads us to a framework in which the specifications of such analog systems are given by fixed points of certain operators on continuous data streams. Such a framework was considered by Tucker and Zucker [TZ07]. We study the properties of these analog systems and their associated operators, and present a characterization of the X-GPAC-generable functions which generalizes Shannon's results.

1 Introduction

Analog computation, as conceived by Kelvin [TT80], Bush [Bus31], and 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, electromechanical or electronic components [Sma93, Hol96, Joh96].

A general purpose analog computer (GPAC) was introduced by Shannon [Sha41] as a model of Bush's Differential Analyzer [Bus31]. 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 Pour-El [PE74] who gave some good characterizations of the analog computable functions, focusing on the classic analog systems built from constants, adders, multipliers and integrators. 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 characterization in terms of algebraic differential equations, Pour-El showed that not all computable functions on the reals (in the sense of computable analysis) can be obtained with these analog networks.

The theory of analog computing has also been developed by Moore with some very general mathematical models [Moo96]. These models, using schemes rather like Kleene's [Kle55], but with primitive recursion replaced by integration and others added, define a hierarchy of functions on the reals, which contains the GPAC generable functions at its lowest level, and uncomputable functions (in the sense of computable analysis) at higher levels. Graça and Costa [GC03] have presented an improved model of the GPAC, and shown this to be equivalent to the lowest level subclass of Moore's functions.

The contributions of Campagnolo [CMC02] and Mycka [MC04] have also presented some fine results concerning analog complexity classes. Finally, Pouly [Pou15] studied in his PhD thesis the GPAC (among other models of computation) from the point of view of complexity classes, and with Bournez and Graça [BGP16] they have defined a multidimensional GPAC (however, their model can be formulated as working under a single, 'implicit time' variable, as will be discussed below).

The main objects of our study are *analog networks* or *analog systems*, [TZ07, TZ11, JZ13, TZ14], whose main components are described as follows:

$$\textit{Analog network} = \textit{data} + \textit{time} + \textit{channels} + \textit{modules}.$$

We will model *data* as elements of a complete metric vector space X, such as a Banach or Fréchet space. We will use a bounded interval of the real numbers as a continuous model of *time* $\mathbb{T} = [0, T]$, where T denotes the final time. Each *channel* carries a continuously differentiable stream, represented as a function $u : \mathbb{T} \to X$ (this space is denoted by $C^1(\mathbb{T}, X)$). Each *module* M has zero, one or more input channels, and must have a single output channel; thus it can be specified by a (possibly partially defined) *stream function*

$$F_M : X^k \times C^1(\mathbb{T}, X)^\ell \rightharpoonup C^1(\mathbb{T}, X).$$

In the case that $X = \mathbb{R}$, we obtain the Shannon GPAC. We can use four types of modules to describe this model, which are equivalent to the ones originally used by Shannon.

Definition 1.1 (**Shannon modules**)**.** The *Shannon modules* are defined as follows:

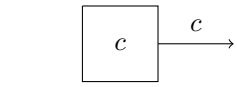
- for each $c \in \mathbb{R}$, there is a *constant module* with zero inputs and one output v, given by
$$v(t) = c;$$
- the *adder module* has two inputs u, v and one output w, given by
$$w(t) = u(t) + v(t);$$
- the *multiplier module* has two inputs u, v and one output w, given by
$$w(t) = u(t)v(t);$$
- the *integrator module* has an initial setting c, two inputs u, v and one output w, given by the Lebesgue-Stieltjes integral
$$w(t) = c + \int_0^t u(s)v'(s)ds;$$

We can depict the four Shannon modules in box diagrams, as in Figure 1. We also introduce the symbol '$\int$' to denote the integrator operator.

The continuous differentiability of the streams u, v ensures that the integrator module is well defined; indeed, the Lebesgue-Stieltjes integral is well defined for continuous integrand and continuously differentiable integrator. In other words, for any $u, v \in C^1([0, T], \mathbb{R})$, the formula $\int_0^t u(s)dv(s)$ defines a function in $C^1([0, T], \mathbb{R})$.

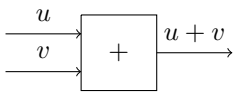
Definition 1.2 (**Shannon GPAC**)**.** A *Shannon general purpose analog computer* (GPAC) is a network built with the four Shannon modules (constants, adders, multipliers and integrators) and connections between their inputs and outputs, with the following restrictions:

- the only connections allowed are between an output and an input;
- each input may be connected to either zero or one output;



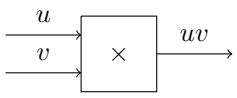
$c :\to C^1([0,T],\mathbb{R})$

$c(t) = c$



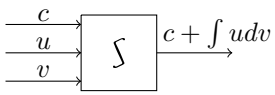
$+ : C^1([0,T],\mathbb{R}) \times C^1([0,T],\mathbb{R}) \to C^1([0,T],\mathbb{R})$

$+(u,v)(t) = u(t) + v(t)$



$\times : C^1([0,T],\mathbb{R}) \times C^1([0,T],\mathbb{R}) \to C^1([0,T],\mathbb{R})$

$\times(u,v)(t) = u(t)v(t)$



$\int : \mathbb{R} \times C^1([0,T],\mathbb{R}) \times C^1([0,T],\mathbb{R}) \to C^1([0,T],\mathbb{R})$

$\int(c,u,v)(t) = c + \int_0^t u(s)dv(s)$

Figure 1: The four Shannon modules.

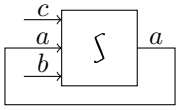


Figure 2: A GPAC for computing the exponential function.

Example 1.3. Of course, there can be cycles in a GPAC, for example, an output connected to an input of its own module. This is called *feedback* and it is fundamental to develop an interesting theory. A simple example of a Shannon GPAC can be seen in Figure 2. It has only one module, which is an integrator module, and only one connection, between its output channel and one of its input channels.

In his paper Shannon sets out to characterize the class of functions which can be generated by a GPAC and in doing so he found an equivalence with the class of differentially algebraic functions.

Definition 1.4 (**Differentially algebraic function**)**.** A function $f : [0,T] \to \mathbb{R}$ is said to be *differentially algebraic* if there exists $k \in \mathbb{N}$ and a polynomial P in $k+2$ variables (and real coefficients) such that $f \in C^k([0,T])$ and

$$P(t, f(t), f'(t), \ldots, f^{(k)}(t)) = 0, \quad \text{for all } t \in [0,T]. \tag{1}$$

An equation in the form (1) is called a *differential algebraic equation.*

Theorem 1 (**GPAC characterization theorem**)**.** *Let $u \in C^1(\mathbb{T},\mathbb{R})$. Then u is generable by a GPAC if and only if u is differentially algebraic.*

The original proof of Theorem 1 can be found in [Sha41], but that proof had flaws, which were corrected in the papers [PE74], [LR87], [GC03].

In this paper we present a generalization of the Shannon GPAC for channels whose values lie in a general complete metric vector space, i.e. a Banach or Fréchet space. This allows us to, for example, work with functions of more than one variable. Our goal is to obtain an equivalence result similar to Theorem 1, with a larger class of functions which include, in particular, solutions to well-known PDEs, such as the heat equation and wave equation.

The paper is organized as follows. In Section 2 we show some limitations to Shannon's model and motivate our transition to function data spaces. In Section 3 we present our new model, the X-GPAC, and define its semantics. In Section 4 we introduce *normal form systems* which are used as an intermediate step towards the main result. In Section 5 we introduce *partial differential algebraic systems* and prove our main result: a characterization of X-GPAC-generable functions in terms of solutions to such systems. In Section 6 we summarize our findings and propose directions for further research.

2 Limitations of the Shannon GPAC

The Shannon GPAC is regarded as an important and powerful method of analog computation, thanks largely to Theorem 1. Despite this, many authors have pointed out some limitations to the model. For example, the gamma function

$$\Gamma(t) = \int_0^\infty x^{t-1} e^{-x} dx$$

is not differentially algebraic, and so cannot be generated by a GPAC (this was noted by Shannon himself). However, one could expect that, in a 'sensible' model of computability on continuous data, this function would be computable. Indeed, the gamma function is *effectively computable* in the sense of computable analysis, a branch of analog computability studied by Pour-El, Richards [PER79], Weihrauch [Wei00], Grzegorczyk [Grz55, Grz57], Lacombe [Lac55a, Lac55b, Lac55c], Tucker, Zucker [TZ07], among others. Some authors have addressed this limitation, and Graça [Gra04] showed that the gamma function can indeed be considered as GPAC computable if

> ...we redefine our notion of GPAC-computability in a manner that it matches more closely the philosophy underlying computable analysis...

There is, however, another limitation with the Shannon GPAC, that appears to have been overlooked by Shannon, Pour-El and others. It lies in the fact that the Shannon GPAC can fundamentally reason only about real-valued functions of one independent variable t. Ironically, it was stated in [Sha41] and [PE74] that the generalization to more than one independent variable only requires an obvious modification, but this is by no means the case. In fact, it is hard to conceive a realistic physical interpretation for a formalism involving two (or more) independent "time" variables.

We briefly remark that this limitation was pointed out in [Rub93]. Rubel says

> For one thing, the GPAC works in one ("time") variable only, while the EAC [Extended Analog Computer] produces functions of any finite number of real variables.

To address this problem, Rubel defined what he called an Extended Analog Computer (EAC).[1] However, Rubel stressed that

> ... the EAC is a *conceptual* computer - the extent to which it can be realized by actual physical, chemical, or biological devices or systems remains to be investigated.

A different attempt to deal with this problem was recently proposed by Bournez, Graça and Pouly [Pou15, BGP16]. In their approach, channels can carry a n-variable real-valued data stream of type $\mathbb{R}^n \to \mathbb{R}$. However, their model can still be re-expressed in terms of only one (implicit "time") variable (see [BGP16, Examples 12 and 13]).

We believe that extending the *input space*, that is, replacing $C^1([0,T],\mathbb{R})$ with $C^1([0,X_1] \times \ldots \times [0,X_n],\mathbb{R})$ may not be the intended direction. Instead, our idea is to extend the *output space*, that is, replacing $C^1([0,T],\mathbb{R})$ with $C^1([0,T],X)$, where X is a metric vector space. For example, we can think of X as the space of continuous real-valued functions on a bounded domain $\Omega \subset \mathbb{R}^n$, that is, $X = C(\Omega,\mathbb{R})$. In this way, our channels will now carry X-valued streams of data $u : [0,T] \to X$, which correspond to functions of $n+1$ real variables, under the uncurrying

$$[0,T] \to (\Omega \to \mathbb{R}) \simeq [0,T] \times \Omega \to \mathbb{R}.$$

It is evident that one of the variables, namely the "time" variable, plays a different role from the others.

[1] An implementation of the EAC (or at least, of some of its components) has been achieved with the work of Mills, [Mil08].

3 The X-GPAC

As discussed in the previous section we decide to change our data space X to become a function space. In this paper we shall thus take X to be the space of continuous real functions of a real variable,

$$X = C(\mathbb{R}).$$

We observe that X is a Fréchet space, with the family of pseudonorms

$$\|g\|_M = \sup_{|x|\leq M} |g(x)|. \tag{2}$$

We also consider the subspace $D = C^1(\mathbb{R})$ of continuously differentiable functions. This is also a Fréchet space, with the family of pseudonorms

$$\|g\|_M = \sup_{|x|\leq M} |g(x)| + \sup_{|x|\leq M} |\partial_x g(x)|. \tag{3}$$

Note that D is a linear and dense (under the X-pseudonorms) subspace of X.

We can define a differential operator as

$$\begin{array}{rccc} \partial_x : & X & \rightharpoonup & X \\ & u(x) & \mapsto & \partial_x u(x). \end{array} \tag{4}$$

We establish some important properties of the differential operator:

- ∂_x is a partial function from X to X, with domain $\mathrm{dom}(\partial_x) = D$;
- ∂_x is linear, that is, for all $u, v \in D$ and $\alpha, \beta \in \mathbb{R}$ we have $\partial_x(\alpha u + \beta v) = \alpha \partial_x u + \beta \partial_x v$;
- ∂_x has dense domain, that is, D is dense in X (under the topology in X induced by its pseudonorms);
- ∂_x has a closed graph, that is, if (u_n) is a sequence in D with $u_n \to u$ and $\partial_x u_n \to v$, then $u \in D$ and $\partial_x u = v$;

We include here the definition of closed operator, which will play a role in our notion of well-posedness.

Definition 3.1. [**Closed operator**] Let X and Y be metric spaces and $A : X \rightharpoonup Y$ a partial function with domain $D(A)$. We say that A is *closed* if its graph is a closed subset of $X \times Y$; in other words, if for all sequences (x_n) in X, $x \in X$ and $y \in Y$ we have

$$\text{if } \left\{ \begin{array}{ll} x_n \to x & \text{as } n \to \infty, \\ x_n \in D(A) & \text{for all } n, \\ Ax_n \to y & \text{as } n \to \infty; \end{array} \right\} \text{ then } x \in D(A) \text{ and } Ax = y.$$

Hence ∂_x is a closed operator on X. We remind the reader that closedness is, in general, a weaker property than continuity. We also remark that both are equivalent in the space of *total linear* operators between Banach spaces (this basic result is known as the Closed Graph Theorem).

Example 3.2. Consider the sequence $a_n(x) = \frac{1}{n}\sin(n^2 x)$, which converges in $X = C(\mathbb{R})$ (that is, in the pseudonorms of X) to 0; for any M, $\|a_n\|_M = \frac{1}{n} \to 0$. Moreover, each $a_n \in D$, and $\partial_x a_n(x) = n\cos(n^2 x)$. Note that, for any M, $\|a_n'\|_M = n$, meaning that the suprema of a_n' grow without bounds. Thus ∂_x is a discontinuous operator.

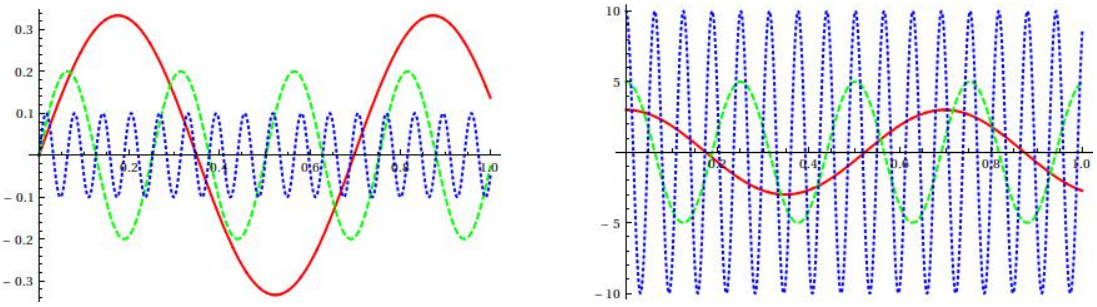


Figure 3: Plot of $a_n(x)$ (left) and $a_n'(x)$ (right) for $n = 3$ (red, bold), $n = 5$ (green, dashed), $n = 10$ (blue, dotted).

We consider X-streams as elements in the space $C^1([0,T],X)$ of X-valued, continuously differentiable functions, for some $T > 0$. By continuous differentiability we mean that, for $u \in C^1([0,T],X)$, the expression

$$v(t) = \lim_{h\to 0} \frac{u(t+h) - u(t)}{h}$$

is well-defined for all $t \in [0,T]$ and v is a continuous function of time.

In this way, $C^1([0,T],X)$ is a Fréchet space under the family of pseudonorms

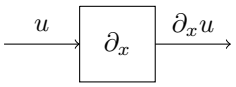
$$\|u\|_{T,M} = \sup_{0\le t\le T} \|u(t)\|_M + \sup_{0\le t\le T} \|u'(t)\|_M;$$

note that in the right-hand side we consider pseudonorms of X.

The Shannon modules are easily defined in this new setting, and we introduce a new module based on the differential operator.

Definition 3.3. The *differential module* has one input u and one output v, given by

$$v(t) = \partial_x u(t);$$



$$\partial_x : C^1([0,T],X) \rightharpoonup C^1([0,T],X)$$

$$\partial_x(u)(t) = \partial_x u(t)$$

Figure 4: The differential module.

We remark that the differential module $\partial_x : C^1([0,T],X) \rightharpoonup C^1([0,T],X)$ is partially defined and its domain is $C^1([0,T],D)$. As mentioned above, ∂_x is a closed but discontinuous operator.

With the above considerations in mind we can arrive at the desired generalization of the Shannon GPAC.

Definition 3.4 (X**-GPAC).** An *X-valued general purpose analog computer* (X-GPAC) is a network built with the five X-modules (constants, adders, multipliers, integrators and differentials) and X-channels connecting their inputs and outputs, with the following restrictions:

- the only connections allowed are between an output and an input;
- each input may be connected to either zero or one output;

Our next goal is to assign semantics to an X-GPAC, that is, determine how to define generable functions. For that end, we introduce the notion of induced operator.

Definition 3.5 (X**-GPAC induced operator).** Let $\mathcal{G}$ be an X-GPAC;

- the *constant space* of $\mathcal{G}$ is the cartesian product of the spaces associated with all the initial settings occuring in any integrator. The constant space can be written as $\mathcal{C} = X^p$, for some $p \ge 0$;
- the *proper input space* of $\mathcal{G}$ is the cartesian product of the spaces associated with all the unconnected input channels. The input space can be written as $\mathcal{I} = C^1([0,T],X)^q$, for some $q \ge 0$;
- the *proper output space* of $\mathcal{G}$ is the cartesian product of the spaces associated with all the unconnected output channels. The output space can be written as $\mathcal{O} = C^1([0,T],X)^m$, for some $m \ge 0$;
- the *mixed space* of $\mathcal{G}$ is the cartesian product of the spaces associated with all the channels which connect some input with some output. The mixed space can be written as $\mathcal{M} = C^1([0,T],X)^r$, for some $r \ge 0$;
- the *induced operator* of $\mathcal{G}$ is the function

$$F : \mathcal{C} \times \mathcal{I} \times \mathcal{M} \rightharpoonup \mathcal{M} \times \mathcal{O}, \quad F(\mathbf{g}, \mathbf{u}^I, \mathbf{u}^M) = (\tilde{\mathbf{u}}^M, \mathbf{u}^O), \tag{5}$$

  where $\mathbf{g}$ is an X-valued vector and each $\mathbf{u}$ is a vector of X-channels. Moreover, each of the components in the codomain of F is given by the module with which it is associated. In other words, for each u_i component of either $\tilde{\mathbf{u}}^M$ or $\mathbf{u}^O$,

– if u_i is associated with the output channel of a constant, then $u_i = g$, where g is the constant associated with that module;
– if u_i is associated with the output channel of an adder, then $u_i = +(v_1, v_2)$, where v_1 and v_2 are the components in $\mathbf{u}^I$ or $\mathbf{u}^M$ associated with the input channels of that module;
– if u_i is associated with the output channel of a multiplier, then $u_i = \times(v_1, v_2)$, where v_1 and v_2 are the components in $\mathbf{u}^I$ or $\mathbf{u}^M$ associated with the input channels of that module;
– if u_i is associated with the output channel of an integrator, then $u_i = \int(g, v_1, v_2)$, where v_1, v_2 are the components in $\mathbf{u}^I$ or $\mathbf{u}^M$ associated with the input channels of that module, and g is the component in $\mathbf{g}$ associated with the initial setting of that module;
– if u_i is associated with the output channel of a differential, then $u_i = \partial_x(v)$, where v is the component in $\mathbf{u}^I$ or $\mathbf{u}^M$ associated with the input channel of that module.

We remark that F may be partially defined; for each differential module occuring in $\mathcal{G}$, there is a component of $\mathcal{I} \times \mathcal{M}$ which is restricted to $C^1([0,T], D)$. Hence, to describe the domain of F we must take into special consideration those channels which are inputs of differential modules.

Example 3.6. To achieve a better understanding of Definition 3.5 we provide an example in Figure 5 of an X-GPAC with one constant and four X-channels.

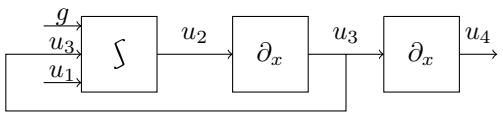


Figure 5: An X-GPAC implementing a transport equation.

In this example there is one constant (associated with the integrator module), one input channel (labeled u_1), two mixed channels (labeled u_2 and u_3) and one output channel (labeled u_4). The induced operator simply formalizes the input/output relation between these channels,

$$\mathcal{C} = X, \quad \mathcal{I} = C^1(\mathbb{T}, X), \quad \mathcal{M} = C^1(\mathbb{T}, X)^2, \quad \mathcal{O} = C^1(\mathbb{T}, X);$$
$$F : \mathcal{C} \times \mathcal{I} \times \mathcal{M} \rightharpoonup \mathcal{M} \times \mathcal{O}$$
$$F(g, u_1, u_2, u_3) = \left(g + \int u_3 du_1, \partial_x u_2, \partial_x u_3\right) = (\tilde{u_2}, \tilde{u_3}, u_4).$$

Definition 3.4 gives a lot of freedom in the construction of X-GPACs and it turns out that not all of the possible networks lead to 'valid and interesting' X-GPACs (similarly to the fact that not all ASCII expressions lead to 'valid and interesting' computer programs). Thus we present a well-posedness-like notion to restrict the space of X-GPACs that we wish to consider.

Definition 3.7 (**Quasi-well-posedness of X-GPAC**). Let $\mathcal{G}$ be an X-GPAC and $F : \mathcal{C} \times \mathcal{I} \times \mathcal{M} \rightharpoonup \mathcal{M} \times \mathcal{O}$ be its induced operator. Let $U \subseteq \mathcal{C} \times \mathcal{I}$. We say that $\mathcal{G}$ is *quasi-well-posed* on U if

- (*existence*) for every $(\mathbf{g}, \mathbf{u}^I) \in U$, there exists $(\mathbf{u}^M, \mathbf{u}^O) \in \mathcal{M} \times \mathcal{O}$ such that

$$F(\mathbf{g}, \mathbf{u}^I, \mathbf{u}^M) = (\mathbf{u}^M, \mathbf{u}^O); \tag{6}$$

- (*uniqueness*) for every $(\mathbf{g}, \mathbf{u}^I) \in U$, the tuple $(\mathbf{u}^M, \mathbf{u}^O)$ such that (6) holds is unique;
- (*closedness*) the map $(\mathbf{g}, \mathbf{u}^I) \mapsto (\mathbf{u}^M, \mathbf{u}^O)$, with domain U and codomain $\mathcal{M} \times \mathcal{O}$, given as the unique solution of (6), defines a closed operator.

We may refer to (6) as the *fixed point equation*; note that the mixed variables $\mathbf{u}^M$ are the only ones that appear in both sides of the equation.

If we required continuity instead of closedness (that is, if we required U to be an open set, and the map $(\mathbf{g}, \mathbf{u}^I) \mapsto (\mathbf{u}^M, \mathbf{u}^O)$ to be continuous), then our definition would match the three usual principles for *well-posedness* - existence, uniqueness, continuity of solutions - as presented by Hadamard [Had52] (see also [CH53]). Instead, we choose to use closedness (and a non-necessarily open domain U), which is a strictly weaker criterion, hence the term 'quasi-well-posed'. The reason for choosing closedness is

the presence of the differential module, which defines a closed function of type $X \rightharpoonup X$ which is not continuous.

Admittedly, some could argue that a discontinuous operation is of little interest in the study of computable functions of continuous spaces. We must then make the important remark that continuity can be obtained from closedness (and thus well-posedness can be obtained by quasi-well-posedness) if we define a finer topology in the domain, induced by graph (pseudo)norms. To be precise, we recall the following basic result in functional analysis.

Proposition 3.8 ([RR06, p. 240, Exercise 8.7])**.** *Let $(X, \|\cdot\|_X)$ and $(Y, \|\cdot\|_Y)$ be Banach spaces and $A : X \rightharpoonup Y$ a closed linear operator with domain $D(A)$. Consider the Banach space $(D(A), \|\cdot\|_{G(A)})$, with the graph norm given by*

$$\|x\|_{G(A)} = \|x\|_X + \|Ax\|_Y.$$

Then the restriction $A : D(A) \to Y$ is a continuous linear map between Banach spaces.

Proof. We first prove that $\|\cdot\|_{G(A)}$ defines a norm:

- if $x \in D(A)$, then $\|x\|_{G(A)} = 0$ iff $\|x\|_X + \|Ax\|_Y = 0$ iff $\|x\|_X = 0$ iff $x = 0$;
- if $x \in D(A)$ and $\alpha \in \mathbb{R}$, then $\|\alpha x\|_{G(A)} = \|\alpha x\|_X + \|A(\alpha x)\|_Y = |\alpha|\|x\|_X + |\alpha|\|Ax\|_Y = |\alpha|\|x\|_{G(A)}$;
- if $x, y \in D(A)$, then $\|x + y\|_{G(A)} = \|x + y\|_X + \|A(x + y)\|_Y \leq \|x\|_X + \|y\|_x + \|Ax\|_Y + \|Ay\|_Y = \|x\|_{G(A)} + \|y\|_{G(A)}$.

Next we prove that $D(A)$ is complete. Let $\{x_n\}_n$ be a Cauchy sequence in $D(A)$, so that $\|x_{N+k} - x_N\|_{G(A)} = \|x_{N+k} - x_N\|_X + \|A(x_{N+k} - x_N)\|_Y$ vanishes (uniformly on k) as $N \to \infty$. In particular, we must have that $\{x_n\}_n$ is a Cauchy sequence in X (under the X-norm) and $\{Ax_n\}$ is a Cauchy sequence in Y. Since X and Y are Banach spaces it follows that there exist $x \in X$ and $y \in Y$ such that $x_n \to x$ and $Ax_n \to y$. By closedness of A we conclude that $x \in D(A)$ and $Ax = y$, so that $x_n \to x$ in $D(A)$ (under the graph norm). Thus $(D(A), \|\cdot\|_{G(A)})$ is a Banach space.

Finally we prove that $A : D(A) \to Y$ is continuous. Let $\{x_n\}_n$ be a sequence in $D(A)$ and $x \in D(A)$ such that $x_n \to x$ in $D(A)$. Then $\|x_n - x\|_{G(A)} \to 0$, which implies $\|x_n - x\|_X \to 0$ and $\|A(x_n - x)\|_Y \to 0$, so that, in particular, $Ax_n \to Ax$ in Y. □

Proposition 3.9. *Let X and Y be Fréchet spaces with families of pseudonorms $\{\|\cdot\|_{X,n}\}_n$ and $\{\|\cdot\|_{Y,m}\}_m$ and $A : X \rightharpoonup Y$ a closed linear operator with domain $D(A)$. Consider the Fréchet space $D(A)$ with graph pseudonorms given by*

$$\|x\|_{G(A),n,m} = \|x\|_{X,n} + \|Ax\|_{Y,m}.$$

Then the restriction $A : D(A) \to Y$ is a continuous linear map between Fréchet spaces.

Proof. The proof is similar to that of Proposition 3.8. □

This finer topology is usually equivalent to a topology of interest in the domain space. For example, consider the differential operator $\partial_x : C(\mathbb{R}) \rightharpoonup C(\mathbb{R})$ given by (4), which is closed but not continuous under the usual family of pseudonorms in $C(\mathbb{R})$. However, it becomes a continuous operator if we restrict it to the space $C^1(\mathbb{R})$ and consider the graph pseudonorms

$$\|f\|_{n,m} = \|f\|_n + \|\partial_x f\|_m = \sup_{|x| \leq n} |f(x)| + \sup_{|x| \leq m} |\partial_x f(x)|,$$

under whose topology $C^1(\mathbb{R})$ is a Fréchet space. Moreover, this family of pseudonorms can be seen to be equivalent to the usual family of pseudonorms in $C^1(\mathbb{R})$ given by (3).

We shall then adopt the notion of quasi-well-posedness in this paper, while reminding ourselves that, if needed, we can in principle express our results in terms of well-posed operators.

The final step in this section is to assign semantics to X-GPACs, that is, to define the notion of X-GPAC-generable functions.

Definition 3.10 (**Semantics of X-GPAC**)**.**

(a) Let $\mathcal{G}$ be an X-GPAC and $F : \mathcal{C} \times \mathcal{I} \times \mathcal{M} \rightharpoonup \mathcal{M} \times \mathcal{O}$ be its induced operator. Let $U \subseteq \mathcal{C} \times \mathcal{I}$ such that $\mathcal{G}$ is quasi-well-posed on U. We define the *specification* of $\mathcal{G}$ as the (partial) function

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

whose domain is U and where $(\mathbf{u}^M, \mathbf{u}^O)$ is given by (6). We also say that $\mathcal{G}$ *generates* Φ on $U = \mathrm{dom}(\Phi)$.

(b) A function $\Phi : \mathcal{C} \times \mathcal{I} \rightharpoonup \mathcal{M} \times \mathcal{O}$ is *X-GPAC-generable* if there exists an X-GPAC $\mathcal{G}$ such that $\mathcal{G}$ is quasi-well-posed on the domain of Φ and Φ is the specification of $\mathcal{G}$.

We remark that every integrator in an X-GPAC has an initial setting (which is one of the constants in the space $\mathcal{C}$) and an output (which is one of the mixed/output channels in $\mathcal{M} \times \mathcal{O}$). Since we can define an injective map from the initial settings to the mixed/output channels, we have the following basic property.

Proposition 3.11. *If $\Phi : \mathcal{C} \times \mathcal{I} \rightharpoonup \mathcal{M} \times \mathcal{O}$ is X-GPAC-generable, then* $\dim(\mathcal{C}) \leq \dim(\mathcal{M}) + \dim(\mathcal{O})$.

4 Normal form systems

We have defined in the previous section the class of X-GPAC-generable functions, which constitute our space of interest. We can state our objective as follows.

Problem. Characterize the class of X-GPAC-generable functions in terms of a suitable generalization of the class of differential algebraic equations.

In the study of the GPAC, an intermediate step is usually taken in the transition from generable functions to differential algebraic equations. For example, in [Sha41] a system of equations called a *fundamental solvability condition* was considered (Theorem I in that paper). Also in [PE74] a similar system of equations is used in the actual definition of GPAC generable functions (Definition 10 in that paper) instead of the usual definition involving analog networks. We shall generalize that notion into our framework, and refer to the resulting objects as normal form systems.

Definition 4.1 (**Normal form equation**). Let $N \in \mathbb{N}$. A *normal form equation* on the N variables $y_1, \ldots, y_N$ is an equation of the form

$$\sum_{i=0}^{N} \sum_{j=1}^{N} b_{ij} y_i y_j' + \sum_{j=1}^{N} c_j \partial_x y_j' = 0, \tag{7}$$

under the conventions that $y_0 \equiv 1$ and b_{ij}, c_j are real numbers.

We remark that $y_0 \equiv 1$ is not a variable, just a notational convenience to obtain a more compact equation. We also note that index i starts at 0 whereas index j starts at 1; thus (7) will not include the term y_0' (which would be equal to 0, and therefore irrelevant).

We also alert the reader to our choice of terms of the form $\partial_x y_j'$ in the second summation in (7), with derivatives both in time and space. One could think that a similar definition of normal form equations with terms of the form $\partial_x y_j$ would be more 'natural'. In fact, both definitions would be equivalent, and we could move from the former to the latter by adding extra variables (namely, one would have to add the extra variable $\mathbf{t}$ that specifies "linear time", $\mathbf{t}(t, x) = t$). The main reason for choosing (7) as our template for normal form equations is of practicality, as it makes the proof of Lemma 4.7 (below) much simpler.

Here are some examples of normal form equations:

$$y_1' = y_1 y_2';$$
$$y_1' = y_2' + y_3';$$
$$y_1' = \partial_x y_1'.$$

Definition 4.2 (**Normal form system over** X)**.** Let $K, L \in \mathbb{N}$ and $N = K + L$. A *normal form system* (NFS) on the N variables $y_1, \ldots, y_N$ is a system of the form

$$\begin{cases} \sum_{i=0}^{N}\sum_{j=1}^{N} b_{ij\ell} y_i y_j' + \sum_{j=1}^{N} c_{j\ell}\partial_x y_j' = 0 & \text{, for } \ell = 1, \ldots, L; \\ y_{K+\ell}(0) = g_\ell & \text{, for } \ell = 1, \ldots, L, \end{cases} \tag{8}$$

under the conventions that $y_0 \equiv 1$, $b_{ij\ell}$, $c_{j\ell}$ are real numbers and $g_\ell \in X$.

We say that $g_1, \ldots, g_L$ are the *(initial) constants*, $y_1, \ldots, y_K$ are the *inputs* or *independent variables* and $y_{K+1} \ldots, y_{K+L}$ are the *outputs* or *dependent variables*.

Definition 4.3 (**Quasi-well-posedness of NFS**)**.** Let $K, L \in \mathbb{N}$ and $N = K + L$. Let $\mathcal{N}$ be an NFS given by (8) with K inputs and L outputs and consider the spaces

$$\mathcal{C} = X^L, \quad \mathcal{I} = C^1([0,T], X)^K, \quad \mathcal{O} = C^1([0,T], X)^L.$$

Let $U \subseteq \mathcal{C} \times \mathcal{I}$. We say that $\mathcal{N}$ is *quasi-well-posed* on U if

- (*existence*) for every $(\mathbf{g}, \mathbf{y}^I) \in U$, there exists $\mathbf{y}^O \in \mathcal{O}$ such that (8) holds for $(\mathbf{g}, \mathbf{y}^I, \mathbf{y}^O)$, where $\mathbf{g} = (g_1, \ldots, g_L)$, $\mathbf{y}^I = (y_1, \ldots, y_K)$, $\mathbf{y}^O = (y_{K+1}, \ldots, y_{K+L})$;
- (*uniqueness*) for every $(\mathbf{g}, \mathbf{y}^I) \in U$, the tuple $\mathbf{y}^O$ such that (8) holds is unique;
- (*closedness*) the map $(\mathbf{g}, \mathbf{y}^I) \mapsto \mathbf{y}^O$, with domain U and codomain $\mathcal{O}$, given as the unique solution of (8), defines a closed operator.

Definition 4.4 (**Semantics of NFS**)**.**

(a) Let $K, L \in \mathbb{N}$ and $N = K + L$. Let $\mathcal{N}$ be an NFS with K inputs and L outputs. Let $U \subseteq \mathcal{C} \times \mathcal{I}$ such that $\mathcal{N}$ is quasi-well-posed on U. We define the *solution* of $\mathcal{N}$ as the (partial) function

$$\begin{aligned} &\Phi : \mathcal{C} \times \mathcal{I} \rightharpoonup \mathcal{O}; \\ &\Phi(\mathbf{g}, \mathbf{y}^I) = \mathbf{y}^O, \end{aligned}$$

whose domain is U and where $\mathbf{y}^O$ is given by (8). We also say that $\mathcal{N}$ *generates* Φ on $U = \mathrm{dom}(\Phi)$.

(b) A function $\Phi : \mathcal{C} \times \mathcal{I} \rightharpoonup \mathcal{O}$ is *NFS-generable* if there exists an NFS $\mathcal{N}$ such that $\mathcal{N}$ is quasi-well-posed on the domain of Φ and Φ is the solution of $\mathcal{N}$.

We remark that in an NFS, there is a bijection between the initial conditions and the outputs; thus we have the following basic property (cf. Proposition 3.11).

Proposition 4.5. *If* $\Phi : \mathcal{C} \times \mathcal{I} \rightharpoonup \mathcal{O}$ *is NFS-generable, then* $\dim(\mathcal{C}) = \dim(\mathcal{O})$.

We have established semantics for NFS, and the next step is to show that every X-GPAC-generable function is a projection of an NFS-generable function, as in the following definition.

Definition 4.6 (**Function projection**)**.** Let $F : A \rightharpoonup B$ and $F' : A \times A' \rightharpoonup B \times B'$, and let

$$\begin{array}{rcccl} G(F) & = & \{(a,b) : a \in \mathrm{dom}(F) \text{ and } F(a) = b\} & \subseteq & A \times B, \\ G(F') & = & \{(a,a',b,b') : (a,a') \in \mathrm{dom}(F') \text{ and } F'(a,a') = (b,b')\} & \subseteq & A \times A' \times B \times B' \end{array}$$

be their graphs. We say that F is a *projection* of F' if for all $(a,b) \in A \times B$,

- if $(a,b) \notin G(F)$ then for all $a' \in A'$, $b' \in B'$ we have $(a,a',b,b') \notin G(F')$;
- if $(a,b) \in G(F)$ then there exist unique $a' \in A'$ and $b' \in B'$ such that $(a,a',b,b') \in G(F')$.

We briefly remark that the notion of projection induces a partial order in the class of functions. We have the following lemma.

Lemma 4.7 (**X-GPAC-generable implies NFS-generable**)**.** *Let* $\Phi_G : \mathcal{C}_G \times \mathcal{I}_G \rightharpoonup \mathcal{M}_G \times \mathcal{O}_G$ *be X-GPAC-generable with domain* $U_G \subseteq \mathcal{C}_G \times \mathcal{I}_G$*. Then there exists* $\Phi_N : \mathcal{C}_N \times \mathcal{I}_N \rightharpoonup \mathcal{O}_N$ *which is NFS-generable with domain* $U_N \subseteq \mathcal{C}_N \times \mathcal{I}_N$ *with the following properties:*

- $\dim(\mathcal{I}_N) = \dim(\mathcal{I}_G)$ *and* $\dim(\mathcal{O}_N) = \dim(\mathcal{M}_G) + \dim(\mathcal{O}_G)$*;*
- *for every* $(\mathbf{g}, \mathbf{u}^I) \in \mathcal{C}_G \times \mathcal{I}_G$ *such that* $(\mathbf{g}, \mathbf{u}^I) \notin U_G$*, we have* $(\mathbf{g}, \mathbf{g}^*, \mathbf{u}^I) \notin U_N$ *for any X-vector* $\mathbf{g}^*$*;*

- *for every* $(\mathbf{g}, \mathbf{u}^I) \in \mathcal{C}_G \times \mathcal{I}_G$ *such that* $(\mathbf{g}, \mathbf{u}^I) \in U_G$*, there exists a unique* X*-vector* $\mathbf{g}^*$ *such that* $(\mathbf{g}, \mathbf{g}^*, \mathbf{u}^I) \in U_N$*; moreover, we have*

$$\Phi_G(\mathbf{g}, \mathbf{u}^I) = \Phi_N(\mathbf{g}, \mathbf{g}^*, \mathbf{u}^I).$$

In other words, Φ_G *is a projection of* Φ_N*.*

Proof. Let $\Phi_G : \mathcal{C}_G \times \mathcal{I}_G \rightharpoonup \mathcal{M}_G \times \mathcal{O}_G$ be X-GPAC-generable with domain $U_G \subseteq \mathcal{C}_G \times \mathcal{I}_G$. Denote by $\mathcal{G}$ the GPAC that generates Φ_G and $F : \mathcal{C}_G \times \mathcal{I}_G \times \mathcal{M}_G \rightharpoonup \mathcal{M}_G \times \mathcal{O}_G$ the induced operator.

The important idea of the proof is understanding how to write the equational specification of F as an NFS. We apply the following conversions, for every u appearing in $\mathcal{M}_G \times \mathcal{O}_G$:

$$\begin{aligned}
u = g \quad &\rightsquigarrow \quad \begin{cases} u' = 0 \\ u(0) = g \end{cases} \\
u = v + w \quad &\rightsquigarrow \quad \begin{cases} u' = v' + w' \\ u(0) = v(0) + w(0) \end{cases} \\
u = v \cdot w \quad &\rightsquigarrow \quad \begin{cases} u' = vw' + wv' \\ u(0) = v(0) \cdot w(0) \end{cases} \\
u = g + \int v dw \quad &\rightsquigarrow \quad \begin{cases} u' = vw' \\ u(0) = g \end{cases} \\
u = \partial_x v \quad &\rightsquigarrow \quad \begin{cases} u' = \partial_x v' \\ u(0) = \partial_x v(0) \end{cases}
\end{aligned}$$

We observe that the input/output relation of each module of $\mathcal{G}$ can be written as a normal form equation coupled with an initial condition. Therefore, we can construct an NFS, $\mathcal{N}$, from $\mathcal{G}$, including an initial condition for each channel.

However, the constant space $\mathcal{C}_G$ appearing in the specification of $\mathcal{G}$ only takes into account those constants appearing as initial settings of integrators. In order to define the solution mapping, Φ_N, we need to extend the constant space to include the initial settings of the other types of operations, as they appear in the list of conversions above.

Let $\mathcal{I}_N = C^1([0,T], X)^K$ and $\mathcal{O}_N = C^1([0,T], X)^L$, where K, L are the number of inputs, outputs of $\mathcal{N}$. By construction each input of $\mathcal{N}$ corresponds to an input channel of $\mathcal{G}$ and each output of $\mathcal{N}$ corresponds to either a mixed or output channel of $\mathcal{G}$. Therefore, we have that $\mathcal{I}_N = \mathcal{I}_G$ and $\mathcal{O}_N = \mathcal{M}_G \times \mathcal{O}_G$, so that $\dim(\mathcal{I}_N) = \dim(\mathcal{I}_G)$ and $\dim(\mathcal{O}_N) = \dim(\mathcal{M}_G) + \dim(\mathcal{O}_G)$, which proves the first bullet.

The next step is to find a suitable U_N for the domain of Φ_N. By quasi-well-posedness of $\mathcal{G}$ on U_G, we know that, for every $(\mathbf{g}, \mathbf{u}^I) \in U_G$, there exists a unique $(\mathbf{u}^M, \mathbf{u}^O) \in \mathcal{M}_G \times \mathcal{O}_G$ such that $F(\mathbf{g}, \mathbf{u}^I, \mathbf{u}^M) = (\mathbf{u}^M, \mathbf{u}^O)$. Thus, if we define $(\mathbf{g}, \mathbf{g}^*)$ as the vector of initial conditions[2] of $(\mathbf{u}^M, \mathbf{u}^O)$, we can infer that $\mathbf{g}^*$ depends uniquely on $(\mathbf{u}^M, \mathbf{u}^O)$, and thus it depends uniquely on $(\mathbf{g}, \mathbf{u}^I)$. With this in mind we define

$$\begin{aligned}
U_N &= \{(\mathbf{g}, \mathbf{g}^*, \mathbf{u}^I) : (\mathbf{g}, \mathbf{u}^I) \in U_G \text{ and } (\mathbf{g}, \mathbf{g}^*) = \Phi_G(\mathbf{g}, \mathbf{u}^I) \restriction_{t=0}\}; \\
\Phi_N(\mathbf{g}, \mathbf{g}^*, \mathbf{u}^I) &= \begin{cases} \Phi_G(\mathbf{g}, \mathbf{u}^I) & \text{if } (\mathbf{g}, \mathbf{g}^*, \mathbf{u}^I) \in U_N; \\ \text{undefined} & \text{otherwise.} \end{cases}
\end{aligned}$$

By the above construction, U_N and Φ_N satisfy the second and third bullets, and all is left is to show that Φ_N is the specification of $\mathcal{N}$ on U_N. By quasi-well-posedness of $\mathcal{G}$ on U_G, it is clear that for $(\mathbf{g}, \mathbf{g}^*, \mathbf{u}^I) \in U_N$, the tuple $(\mathbf{u}^M, \mathbf{u}^O) \in \mathcal{O}_N$ that solves the NFS exists, is unique and given by $\Phi_G(\mathbf{g}, \mathbf{u}^I)$.

To prove closedness of Φ_N, consider a sequence $(\mathbf{g}_n, \mathbf{g}_n^*, \mathbf{u}_n^I) \in U_N$ such that

$$(\mathbf{g}_n, \mathbf{g}_n^*, \mathbf{u}_n^I) \to (\mathbf{g}, \mathbf{g}^*, \mathbf{u}^I) \text{ and } \Phi_N(\mathbf{g}_n, \mathbf{g}_n^*, \mathbf{u}_n^I) \to (\mathbf{u}^M, \mathbf{u}^O);$$

then we have

[2]Possibly after reordering the mixed and output channels; this can be done without loss of generality.

$$(\mathbf{g}_n, \mathbf{u}_n^I) \to (\mathbf{g}, \mathbf{u}^I) \text{ and } \Phi_G(\mathbf{g}_n, \mathbf{u}_n^I) = \Phi_N(\mathbf{g}_n, \mathbf{g}_n^*, \mathbf{u}_n^I) \to (\mathbf{u}^M, \mathbf{u}^O).$$

By closedness of Φ_G, we have

$$(\mathbf{g}, \mathbf{u}^I) \in U_N \text{ and } \Phi_G(\mathbf{g}, \mathbf{u}^I) = (\mathbf{u}^M, \mathbf{u}^O);$$

Now define $(\mathbf{u}_n^M, \mathbf{u}_n^O) = \Phi_G(\mathbf{g}_n, \mathbf{u}_n^I)$, so that

$$(\mathbf{g}_n, \mathbf{g}_n^*) = \Phi_G(\mathbf{g}_n, \mathbf{u}_n^I) \restriction_{t=0} = (\mathbf{u}_n^M, \mathbf{u}_n^O) \restriction_{t=0};$$

by taking limits, we conclude that $(\mathbf{g}, \mathbf{g}^*) = (\mathbf{u}^M, \mathbf{u}^O) \restriction_{t=0}$. Thus,

$$(\mathbf{g}, \mathbf{g}^*, \mathbf{u}^I) \in U_N \text{ and } \Phi_N(\mathbf{g}, \mathbf{g}^*, \mathbf{u}^I) = (\mathbf{u}^M, \mathbf{u}^O),$$

which concludes the proof. □

Example 4.8. In order to better understand the construction in the proof of Lemma 4.7, we apply it to the X-GPAC seen on Example 3.6 (repeated in Figure 6).

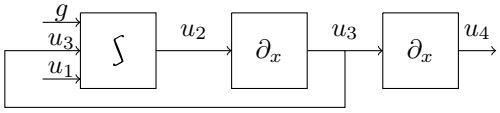


Figure 6: An X-GPAC implementing a transport equation.

It can be checked that this X-GPAC is well-posed for $u_1 \in C^1([0,T], X)$ and $g \in C^2(\mathbb{R})$. It generates the function $\Phi_G : X^1 \times C^1([0,T], X)^1 \rightharpoonup C^1([0,T], X)^3$, given by $\Phi_G(g, u_1) = (u_2, u_3, u_4)$, where

$$u_2(t,x) = g(x + u_1(t) - u_1(0)), \quad u_3(t,x) = g'(x + u_1(t) - u_1(0)), \quad u_4(t,x) = g''(x + u_1(t) - u_1(0));$$

if u_1 is linear time, $u_1(t,x) = t$, this has the simpler form

$$u_2(t,x) = g(x+t), \quad u_3(t,x) = g'(x+t), \quad u_4(t,x) = g''(x+t).$$

Let us construct an NFS with solution Φ_N such that Φ_G is a projection of Φ_N. The X-GPAC generates the equational relations

$$u_2 = g + \int_0^t u_3 du_1, \quad u_3 = \partial_x u_2, \quad u_4 = \partial_x u_3; \tag{9}$$

which can be converted into an NFS with three constants, one input and three outputs,

$$\begin{array}{lll} u_2' = u_3 u_1', & u_3' = \partial_x u_2', & u_4' = \partial_x u_3', \\ u_2(0) = g, & u_3(0) = g_3, & u_4(0) = g_4. \end{array} \tag{10}$$

We can see that the constant space of the NFS has three parameters g, g_3 and g_4, which is two more than in the constant space of the X-GPAC. Therefore any solution of the NFS must be of type $X^3 \times C^1([0,T], X)^1 \rightharpoonup C^1([0,T], X)^3$.

We also see that if (g, u_1, u_2, u_3, u_4) produces a specification of the X-GPAC, then (u_1, u_2, u_3, u_4) produces a solution of the NFS for the initial conditions g, $g_3 = u_3(0)$, $g_4 = u_4(0)$; in other words, the NFS generates a function Φ_N such that

$$\text{if } \Phi_G(g, u_1) = (u_2, u_3, u_4), \text{ then } \Phi_N(g, u_3(0), u_4(0), u_1) = (u_2, u_3, u_4).$$

Thus Φ_G is a projection of Φ_N, as expected.

5 Partial differential algebraic equations

In this section we will define partial differential algebraic systems, which will prove to be the correct generalization of differentially algebraic equations for our purposes, as will be made clear from our main result (Theorem 2).

Definition 5.1 (**Partial differential algebraic equation**)**.** Let $N \in \mathbb{N}$. A *partial differential algebraic equation* (PDAE) on the N variables $y_1, \dots, y_N$ is an equation of the form

$$P(t, y_1, \dots, y_N, \dots, \partial_x^{\alpha_1} y_1^{(\beta_1)}, \dots, \partial_x^{\alpha_N} y_N^{(\beta_N)}) = 0, \tag{11}$$

where P is a polynomial in $y_1, \dots, y_n$ and some of their derivatives, with real coefficients.

Definition 5.2 (**System of PDAEs**)**.** Let $K, L \in \mathbb{N}$ and $N = K + L$. A *partial differential algebraic system* (PDAS), also referred to as *system of PDAEs*, on the N variables $y_1, \dots, y_N$ is a system of the form

$$\begin{cases} P_\ell(t, y_1, \dots, y_N, \dots, \partial_x^{\alpha_1} y_1^{(\beta_1)}, \dots, \partial_x^{\alpha_N} y_N^{(\beta_N)}) = 0 & \text{, for } 1 \le \ell \le L; \\ y_\ell^{(\beta)}(0) = g_{\ell,\beta} & \text{, for } K+1 \le \ell \le K+L \text{ and } 0 \le \beta < \beta_\ell, \end{cases} \tag{12}$$

under the conventions that P_ℓ are polynomials in $y_1, \dots, y_N$ and some of their derivatives, with real coefficients, and $g_{\ell,\beta} \in X$.

We say that $y_1, \dots, y_K$ are the *inputs* or *independent variables* and $y_{K+1} \dots, y_{K+L}$ are the *outputs* or *dependent variables*.

We provide a short explanation on the notation in the previous definition. Each variable y_i can appear in the polynomial expressions with space derivatives of order at most α_i and time derivatives of order at most β_i; for each y_ℓ which is an output, we need to provide β_ℓ initial conditions; they correspond to the values of y_ℓ and its space derivatives $y_\ell^{(\beta)}$ of order up to $\beta_\ell - 1$ at time $t = 0$,

$$y_\ell(0), \quad y_\ell'(0), \quad \dots, \quad y_\ell^{(\beta_\ell - 1)}(0).$$

This is a standard assumption in the theory of PDEs and is a necessary condition for well-posedness (with fewer initial conditions, the system is underdetermined; with more initial conditions, the system is overdetermined).

Definition 5.3 (**Quasi-well-posedness of PDAS**)**.** Let $L, K \in \mathbb{N}$ and $N = L + K$. Let $\mathcal{P}$ be a PDAS given by (12) with K inputs, L outputs and J initial conditions and consider the spaces

$$\mathcal{C} = X^J, \quad \mathcal{I} = C^1([0,T], X)^K, \quad \mathcal{O} = C^1([0,T], X)^L.$$

Let $U \subseteq \mathcal{C} \times \mathcal{I}$. We say that $\mathcal{P}$ is *quasi-well-posed* on U if

- (*existence*) for every $(\mathbf{g}, \mathbf{y}^I) \in U$, there exists $\mathbf{y}^O \in \mathcal{O}$ such that (12) holds for $(\mathbf{g}, \mathbf{y}^I, \mathbf{y}^O)$, where $\mathbf{g}$ is the vector of initial conditions, $\mathbf{y}^I = (y_1, \dots, y_K)$, $\mathbf{y}^O = (y_{K+1}, \dots, y_{K+L})$;
- (*uniqueness*) for every $(\mathbf{g}, \mathbf{y}^I) \in U$, the tuple $\mathbf{y}^O$ such that (12) holds is unique;
- (*closedness*) the map $(\mathbf{g}, \mathbf{y}^I) \mapsto \mathbf{y}^O$, with domain U and codomain $\mathcal{O}$, given as the unique solution of (12), defines a closed operator.

Definition 5.4 (**Semantics of PDAS**)**.**

(a) Let $K, L \in \mathbb{N}$ and $N = K + L$. Let $\mathcal{P}$ be a PDAS with K inputs and L outputs. Let $U \subseteq \mathcal{C} \times \mathcal{I}$ such that $\mathcal{P}$ is quasi-well-posed on U. We define the *solution* of $\mathcal{P}$ as the (partial) function

$$\begin{aligned} &\Phi : \mathcal{C} \times \mathcal{I} \rightharpoonup \mathcal{O}; \\ &\Phi(\mathbf{g}, \mathbf{y}^I) = \mathbf{y}^O, \end{aligned}$$

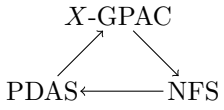
whose domain is U and where $\mathbf{y}^O$ is given by (12). We also say that $\mathcal{P}$ *generates* Φ on $U = \mathrm{dom}(\Phi)$.

(b) A function $\Phi : \mathcal{C} \times \mathcal{I} \rightharpoonup \mathcal{O}$ is *PDAS-generable* if there exists a PDAS $\mathcal{P}$ such that $\mathcal{P}$ is quasi-well-posed on the domain of Φ and Φ is the solution of $\mathcal{P}$.

We remark that in a PDAS, there is a correspondence between the outputs and a subset of the initial conditions (namely, those for which $\beta = 0$), and so we have the following basic property (cf. Propositions 3.11 and 4.5).

Proposition 5.5. *If $\Phi : \mathcal{C} \times \mathcal{I} \rightharpoonup \mathcal{O}$ is PDAS-generable, then $\dim(\mathcal{O}) \leq \dim(\mathcal{C})$.*

Our next two results will complete the cycle



showing that generable functions in each mode can be seen as projections of generable functions in the other modes.

Lemma 5.6 (**NFS-generable implies PDAS-generable**). *Any NFS-generable function is PDAS-generable.*

Proof. Any normal form equation is a partial differential algebraic equation where the variables occur with time (and space) derivatives of order at most 1; thus the initial conditions in an NFS are exactly those appearing as initial conditions in the corresponding PDAS; in other words, every NFS is a PDAS. □

Lemma 5.7 (**PDAS-generable implies X-GPAC-generable**). *Let $\Phi_P : \mathcal{C}_P \times \mathcal{I}_P \rightharpoonup \mathcal{O}_P$ be PDAS-generable with domain $U_P \subseteq \mathcal{C}_P \times \mathcal{I}_P$. Then there exists $\Phi_G : \mathcal{C}_G \times \mathcal{I}_G \rightharpoonup \mathcal{M}_G \times \mathcal{O}_G$ which is X-GPAC-generable with domain $U_G \subseteq \mathcal{C}_G \times \mathcal{I}_G$ with the following properties:*

- $\dim(\mathcal{I}_G) = \dim(\mathcal{I}_P) + 1$ *and* $\dim(\mathcal{C}_G) \geq \dim(\mathcal{C}_P)$*;*
- *for every $(\mathbf{g}, \mathbf{y}^I) \in \mathcal{C}_P \times \mathcal{I}_P$ such that $(\mathbf{g}, \mathbf{y}^I) \notin U_P$, we have $(\mathbf{g}, \mathbf{g}^*, \mathbf{y}^I, y) \notin U_G$ for any X-vector $\mathbf{g}^*$ and any X-stream y;*
- *for every $(\mathbf{g}, \mathbf{y}^I) \in \mathcal{C}_P \times \mathcal{I}_P$ such that $(\mathbf{g}, \mathbf{y}^I) \in U_P$, there exists a unique X-vector $\mathbf{g}^*$ and X-stream y such that $(\mathbf{g}, \mathbf{g}^*, \mathbf{y}^I, y) \in U_N$; moreover, there exists a unique X-stream vector $\mathbf{y}^*$ such that*

$$\Phi_G(\mathbf{g}, \mathbf{g}^*, \mathbf{y}^I, y) = (\mathbf{y}^*, \Phi_P(\mathbf{g}, \mathbf{y}^I)).$$

In other words, Φ_P is a projection of Φ_G.

The equality and inequality in the first bullet will be explained below.

Proof. Let $\Phi_P : \mathcal{C}_P \times \mathcal{I}_P \rightharpoonup \mathcal{O}_P$ be PDAS-generable with domain $U_P \subseteq \mathcal{C}_P \times \mathcal{I}_P$. Denote by $\mathcal{P}$ the PDAS that generates Φ_P.

The important idea of the proof is understanding how to write partial differential algebraic equations with X-GPAC modules. We start with channels for all variables $y_1, \ldots, y_N$. To obtain derivatives in time, we include modules and connections as in Figure 7.

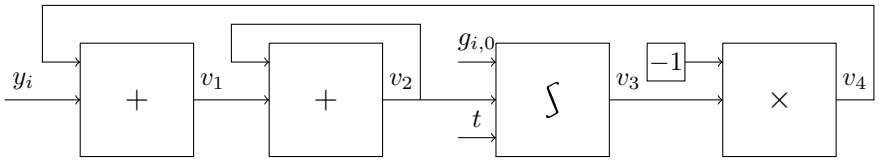


Figure 7: An X-GPAC for computing time derivatives.

Note that the channels on Figure 7 must obey the system

$$\begin{cases} v_1 = y_i + v_4; \\ v_2 = v_1 + v_2; \\ v_3 = g_{i,0} + \int v_2 dt; \\ v_4 = -v_3; \\ y_i(0) = g_{i,0}; \end{cases} \Rightarrow \begin{cases} v_1 = 0; \\ v_2 = y_i'; \\ v_3 = y_i; \\ v_4 = -y_i; \end{cases}$$

so that we obtain y_i' in the channel labeled v_2. Observe that we must include an initial setting (in the above case, the initial value $y_i(0)$) every time we need to take a time derivative. We also need to include the linear channel $\mathbf{t} \in C^1([0,T],X)$ given by the map $(t,x) \mapsto t$. By composing several of these modules, we can get all time derivatives that appear in $\mathcal{P}$, that is, we obtain $y_i^{(\beta)}$ for $i = 1,\dots,N$ and $\beta = 0,\dots,\beta_i - 1$.

Next, by successive applications of the differential module (see Figure 8) we obtain all the partial derivatives appearing in $\mathcal{P}$, which are of the form $\partial_x^\alpha y_i^{(\beta)}$ for $i = 1,\dots,N$, $\alpha = 0,\dots,\alpha_i$ and $\beta = 0,\dots,\beta_i - 1$.

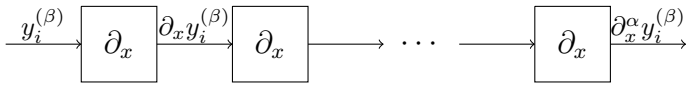


Figure 8: An X-GPAC for computing spatial derivatives.

Next, we compute the polynomial expressions P_ℓ from the partial derivative terms using constants, multipliers and adders. Any term appearing in the polynomials P_ℓ is of the form $at_1^{d_1}\cdots t_n^{d_n}$, where $a \in \mathbb{R}$ (obtainable from a constant module), each t_i is either t (obtainable using the linear input $\mathbf{t}$) or some $\partial_x^\alpha y_i^{(\beta)}$, and thus can be obtained by a constant module and a sequence of multipliers. Each polynomial P_ℓ is a finite sum of such terms, and thus can be obtained by a sequence of adders.

Finally, to enforce the relation $P_\ell(t, y_1,\dots,y_N,\dots,\partial_x^{\alpha_1} y_1^{(\beta_1)},\dots,\partial_x^{\alpha_N} y_N^{(\beta_N)}) = 0$, we add $y_{K+\ell}$ to both sides of the equation and include an adder and feedback loop, as shown on Figure 9. We can then loop the channel labeled $y_{K+\ell}$ back to the start, enforcing it to be a mixed channel.

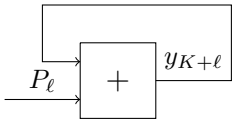


Figure 9: Feedback loop implementing $P_\ell = 0$.

Figure 10 illustrates the several steps of our construction, which results in an X-GPAC $\mathcal{G}$. We must next address the underlying spaces of $\mathcal{G}$. The constant space is constructed with the initial settings of integrators, which only appear in the phase where we build time derivatives. For every original variable y_i which appears in $\mathcal{P}$ with time derivatives of order up to β_i, we have included the β_i initial settings $g_{i,\beta} = y_i^{(\beta)}(0)$, $0 \le \beta < \beta_i$. Hence, the constant space of $\mathcal{G}$, denoted by $\mathcal{C}_G$, is an extension of $\mathcal{C}_P$, which only takes into account the initial settings associated with the output variables; therefore $\dim(\mathcal{C}_G) \ge \dim(\mathcal{C}_P)$.

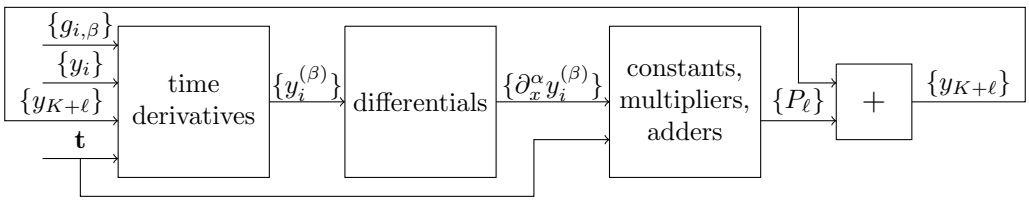


Figure 10: Construction of X-GPAC from a PDAS.

In regards to the input space, the original input variables y_i, $i = 1,\dots,L$ appear in $\mathcal{G}$ as proper input channels. The only other proper input channel is the linear input $\mathbf{t}$, which can be seen as an 'explicit time' inserted into the system. In this way, $\dim(\mathcal{I}_G) = \dim(\mathcal{I}_P) + 1$, which proves the first bullet.

Observe that the original output variables $y_{K+\ell}$, $\ell = 1,\dots,L$, all appear in $\mathcal{G}$ as mixed channels because of the feedback loop that ensures $P_\ell = 0$. There are (potentially many) other mixed and proper output channels in $\mathcal{G}$, which carry the value of either partial derivatives $\partial_x^\alpha y_i^{(\beta)}$, multiplying terms $at_i^{d_i}\cdots t_n^{d_n}$ or sums of multiplying terms; hence $\dim(\mathcal{M}_G \times \mathcal{O}_G) \ge \dim(\mathcal{O}_P)$.

The next step is to find a suitable U_G for the domain of Φ_G. By quasi-well-posedness of $\mathcal{P}$, we know that, for every $(\mathbf{g}, \mathbf{y}^I) \in U_P$, there exists a unique $\mathbf{y}^O \in \mathcal{O}_P$ that solves $\mathcal{P}$. Thus, if we define $(\mathbf{g}^*, \mathbf{g})$ as the vector of initial conditions of all time derivatives of the input and output variables, which are of the form $y_i^{(\beta)}(0)$ for $i = 1, \ldots, N$ and $\beta = 0, \ldots, \beta_i - 1$, we can infer that $\mathbf{g}^*$ depends uniquely on $(\mathbf{y}^I, \mathbf{y}^O)$, and thus it depends uniquely on $(\mathbf{g}, \mathbf{y}^I)$. With this in mind we define

$$U_G = \{(\mathbf{g}^*, \mathbf{g}, \mathbf{y}^I, \mathbf{t}) : (\mathbf{g}, \mathbf{y}^I) \in U_P \text{ and } (\mathbf{g}^*, \mathbf{g}) = (y_1, \ldots, y_1^{(\beta_1-1)}, \ldots, y_N, \ldots, y_N^{(\beta_N-1)}) \restriction_{t=0},$$
$$\text{where } (y_1, \ldots, y_K) = \mathbf{y}^I \text{ and } (y_{K+1}, \ldots, y_{K+L}) = \mathbf{y}^O = \Phi_P(\mathbf{g}, \mathbf{y}^I)\}.$$

To define Φ_G, we need to define the value of all the mixed and output channels appearing in $\mathcal{G}$. As described above, these are either the output variables $y_{K+1}, \ldots, y_{K+L}$, or uniquely obtained from the tuple $(\mathbf{y}^I, \mathbf{y}^O, \mathbf{t}) = (y_1, \ldots, y_K, y_{K+1}, \ldots, y_{K+L}, \mathbf{t})$ via partial derivatives, products and sums. If we let $\mathbf{y}^*$ denote the value of all the mixed and output channels which are not the output variables, then $\mathbf{y}^*$ depends uniquely on $(\mathbf{y}^I, \mathbf{y}^O, \mathbf{t})$ and thus also on $(\mathbf{g}, \mathbf{y}^I)$, so that we can define

$$\Phi_G(\mathbf{g}^*, \mathbf{g}, \mathbf{y}^I, \mathbf{t}) = \begin{cases} (\mathbf{y}^*, \Phi_P(\mathbf{g}, \mathbf{y}^I)) & \text{if } (\mathbf{g}, \mathbf{g}^*, \mathbf{y}^I, \mathbf{t}) \in U_G; \\ \text{undefined} & \text{otherwise.} \end{cases}$$

By this construction, U_G and Φ_G satisfy the second and third bullets, and all is left is to show that Φ_G is the specification of $\mathcal{G}$ on U_G. By quasi-well-posedness of $\mathcal{P}$ on U_P and the above discussion, it is clear that for $(\mathbf{g}^*, \mathbf{g}, \mathbf{y}^I, \mathbf{t}) \in U_G$, the tuple $(\mathbf{y}^*, \mathbf{y}^O) \in \mathcal{O}_G$ that solves the equational specification of $\mathcal{G}$ exists and is unique; that is to say, $\mathbf{y}^O$ is given by $\Phi_P(\mathbf{g}, \mathbf{y}^I)$ and $\mathbf{y}^*$ is obtained from $(\mathbf{y}^I, \mathbf{y}^O, \mathbf{t})$ via partial derivatives, produts and sums.

To prove closedness of Φ_G, consider a sequence[3] $(\mathbf{g}_n^*, \mathbf{g}_n, \mathbf{y}_n^I, \mathbf{t}) \in U_N$ such that

$$(\mathbf{g}_n^*, \mathbf{g}_n, \mathbf{y}_n^I, \mathbf{t}) \to (\mathbf{g}^*, \mathbf{g}, \mathbf{y}^I, \mathbf{t}) \text{ and } \Phi_G(\mathbf{g}_n^*, \mathbf{g}_n, \mathbf{y}_n^I, \mathbf{t}) \to (\mathbf{y}^*, \mathbf{y}^O);$$

by defining $(\mathbf{y}_n^*, \mathbf{y}_n^O) = \Phi_G(\mathbf{g}_n^*, \mathbf{g}_n, \mathbf{y}_n^I, \mathbf{t})$, we have

$$(\mathbf{g}_n, \mathbf{y}_n^I) \to (\mathbf{g}, \mathbf{y}^I) \text{ and } \Phi_P(\mathbf{g}_n, \mathbf{u}_n^I) = \mathbf{y}_n^O \to \mathbf{y}^O.$$

By closedness of Φ_P, we have

$$(\mathbf{g}, \mathbf{y}^I) \in U_P \text{ and } \Phi_P(\mathbf{g}, \mathbf{y}^I) = \mathbf{y}^O.$$

Now $(\mathbf{g}_n^*, \mathbf{g}_n)$ corresponds to the values of the variables in $(\mathbf{y}_n^I, \mathbf{y}_n^O)$ and some of their derivatives at $t = 0$. By closedness of the differential operator, we can take limits and conclude that $(\mathbf{g}^*, \mathbf{g})$ corresponds to the values of the variables in $(\mathbf{y}^I, \mathbf{y}^O)$ and their derivatives at $t = 0$; thus, $(\mathbf{g}^*, \mathbf{g}, \mathbf{y}^I, \mathbf{t}) \in U_G$. Also, since partial derivatives, products and sums are closed operators, we infer that $(\mathbf{y}_n^*, \mathbf{y}_n^O) \to \Phi_G(\mathbf{g}^*, \mathbf{g}, \mathbf{y}^I, \mathbf{t})$, so that $\Phi_G(\mathbf{g}^*, \mathbf{g}, \mathbf{y}^I, \mathbf{t}) = (\mathbf{y}^*, \mathbf{y}^O)$, which concludes the proof. □

Example 5.8. We provide an example of the construction in the previous Lemma by applying it to the one-dimensional heat equation, which can be written as the following PDAS,

$$u' = \partial_x^2 u, \quad u(0) = x.$$

To produce a solution to the heat equation, we consider, for example, a square-integrable initial condition $g \in C(\mathbb{R}) \cap L^2(\mathbb{R})$, for which the solution can be obtained via the heat kernel, that is, $\Phi_P : X \rightharpoonup C^1([0, T], X)$ is given by $\Phi_P(g) = u$, where

$$u(t, x) = \int_{\mathbb{R}} K(t, x - y) g(y) dy, \quad K(t, x) = \frac{1}{\sqrt{4\pi t}} e^{-x^2/4t}.$$

Let us construct an X-GPAC with specification Φ_G such that Φ_P is a projection of Φ_G. We start with a channel for the variable u. Using the constructions on Figures 7 and 8 we obtain channels with the derivatives u', $\partial_x u$ and ∂_x^2. We can then construct the partial differential algebraic expression $P(u, u', \partial_x u, \partial_x^2 u) = u' - \partial_x^2 u$ using one constant module, one multiplier module and one adder module.

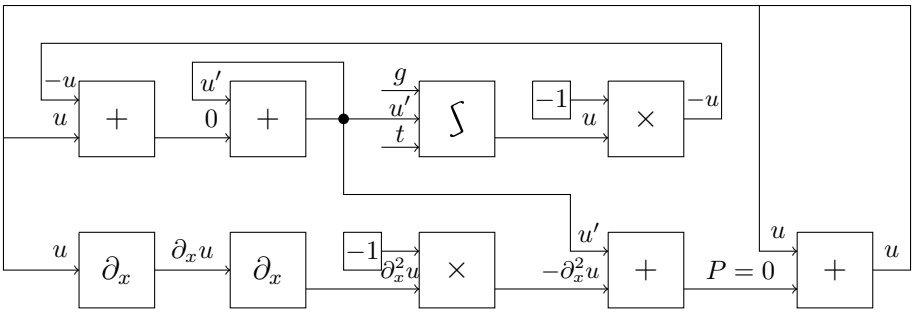


Figure 11: Construction of X-GPAC from the heat equation.

Finally, we include an adder and feedback loop as on Figure 9 and loop the variable u to the beginning. The final X-GPAC can be seen on Figure 11.

We can see that the X-GPAC has one additional input $\mathbf{t}$, which specifies linear time. It should also be noted that the heat equation has only one variable u, whereas the X-GPAC has a total of twelve channels and thus twelve variables (of course, most of those are redundant). Therefore, any specification of the X-GPAC must be of type $X^1 \times C^1([0,T],X)^1 \rightharpoonup C^1([0,T],X)^{11}$.

We can also see that if $\Phi_P(g) = u$ produces a solution to the heat equation, then there is a unique tuple $\mathbf{u}$ of values that satisfy $F(g,\mathbf{t},\mathbf{u}) = \mathbf{u}$, where F is the induced operator of the X-GPAC in Figure 11, g is the initial condition of the integrator and $\mathbf{t}$ is the input channel (linear time). In other words, we can construct a specification $\Phi_G(g,\mathbf{t}) = \mathbf{u}$ of the X-GPAC such that Φ_P is a projection of Φ_G, as expected.

It should be clear from this example that the construction in Lemma 5.7 is universal (in the sense that it works for any PDAS) but is not necessarily optimal (in the sense that it does not add a minimal number of new variables). In fact, one could construct a much simpler X-GPAC (with only three modules!) that generates solutions to the heat equation, as in Figure 12. The reader can verify that Φ_P is also a projection of the specification of this X-GPAC.

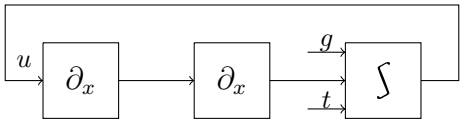


Figure 12: An X-GPAC that generates solutions to the heat equation.

Finally we can state and prove our main result.

Theorem 2 (**X-GPAC Characterization Theorem**). *Let $\Phi : X^j \times C^1([0,T],X)^k \rightharpoonup C^1([0,T],X)^\ell$. Then the following are equivalent:*

- *Φ is the projection of an X-GPAC-generable function;*
- *Φ is the projection of an NFS-generable function;*
- *Φ is the projection of a PDAS-generable function;*

Proof. We prove each implication:

(X-GPAC $\Rightarrow$ NFS) Let Φ be the projection of an X-GPAC-generable function Φ_G. By Lemma 4.7, Φ_G is the projection of an NFS-generable function Φ_N, and so Φ is a projection of Φ_N.

(NFS $\Rightarrow$ PDAS) Similar to the previous case, but use Lemma 5.6 instead.

(PDAS $\Rightarrow$ X-GPAC) Similar to the previous case, but use Lemma 5.7 instead. □

[3]Since the last component of any tuple in U_G must be the linear input $\mathbf{t}$, we only need to consider sequences for the other three subtuples.

6 Conclusion

In this paper we have seen a model of analog computation based on the Shannon GPAC. In this model we have taken streams whose value lie in a general function space X. Theorem 2 is evidence that our model of computation provides a suitable generalization of the original work on the GPAC. We can thus think of solutions to certain differential equations as outputs or fixed points of certain analog networks. We have also seen that (quasi)well-posedness conditions play an important role in this study.

We have only considered the case $X = C(\mathbb{R})$. A possible direction for research would be considering other function spaces, such as

$$X = C^p(\Omega) \quad \text{or} \quad X = H^p(\Omega),$$

where Ω is, for example, a domain in $\mathbb{R}^n$, either unbounded (for example, $\Omega = \mathbb{R}^n$) or bounded (for example, $\Omega = [0,1]^n$), and H^p denotes Sobolev spaces. In the case that Ω is bounded, we may further restrict our space X to functions with prescribed behaviour on the boundary, such as Dirichlet boundary conditions ($f = 0$ on $\partial\Omega$). We believe that such a direction could allow us to make interesting connections with the field of partial differential equations, where such spaces are ubiquitous.

Another possible direction would be to consider modules operating on multiple data types, which could be written as

$$F : \tau_1^{\ell_1} \times \ldots \times \tau_n^{\ell_n} \to \tau;$$

in this way we would have channels of different types and would be able to define a notion of *many-typed analog networks*. This direction will probably have a strongly technical aspect, but it could lead to a model of computation on many-sorted algebras as studied by Tucker and Zucker [TZ00, TZ07, TZ14].

As an interesting question we may wonder whether our framework allows computability of functions which are known not to be Shannon GPAC-generable, such as the gamma function (as noted by Shannon)

$$\Gamma(t) = \int_0^\infty x^{t-1} e^{-x} dx.$$

There are well-known differential equations in two variables related to the gamma function; for example (see [OLBC10, p. 174]), if we define the incomplete gamma functions

$$\gamma(t,z) = \int_0^z x^{t-1} e^{-x} dx;$$

$$\Gamma(t,z) = \int_z^\infty x^{t-1} e^{-x} dx,$$

then both incomplete gamma functions satisfy the differential equation (for $w = w(t,z)$)

$$\frac{d^2 w}{dz^2} + \left(1 + \frac{1-t}{z}\right) \frac{dw}{dz} = 0$$

and we have in addition

$$\gamma(t,z) + \Gamma(t,z) = \Gamma(t), \quad \gamma(t,0) = 0, \quad \Gamma(t,\infty) = 0;$$

we may ask whether such relations could be implemented on a more general analog network.

Acknowledgements. The research of Diogo Poças and Jeffery Zucker was funded, respectively, by Fundação para a Ciência e Tecnologia (Doctoral Grant, Portugal) and the Natural Sciences and Engineering Research Council (Canada).

References

[BGP16] Olivier Bournez, Daniel Graça, and Amaury Pouly. On the functions generated by the general purpose analog computer. submitted on 21 Jan 2016, arXiv:1602.00546, 2016.

[Bus31] Vannevar Bush. The differential analyzer. a new machine for solving differential equations. *Journal of the Franklin Institute*, 212(4):447–488, 1931.

[CH53] R. Courant and D. Hilbert. *Methods of Mathematical Physics*, volume 2. Interscience Publishers, Inc., 1953.

[CMC02] Manuel Campagnolo, Cris Moore, and José Félix Costa. An analog characterization of the Grzegorczyk hierarchy. *Journal of Complexity*, 18(4):977–1000, 2002.

[GC03] Daniel Graça and José Félix Costa. Analog computers and recursive functions over the reals. *Journal of Complexity*, 19(5):644–664, 2003.

[Gra04] Daniel Graça. Some recent developments on shannon's general purpose analog computer. *Mathematical Logic Quarterly*, 50(4–5):473–485, 2004.

[Grz55] A. Grzegorczyk. Computable functions. *Fundamenta Mathematicae*, 42:168–202, 1955.

[Grz57] A. Grzegorczyk. On the defintions of computable real continuous functions. *Fundamenta Mathematicae*, 44:61–71, 1957.

[Had52] Jacques Hadamard. *Lectures on Cauchy's problem in linear partial differential equations*. Dover, 1952.

[Har50] Douglas R. Hartree. *Calculating instruments and machines*. Cambridge University Press, 1950.

[Hol96] Per A. Holst. Svein Rosseland and the Oslo Analyzer. *IEEE Annals of the History of Computing*, 18(4):16–26, 1996.

[Joh96] Magnus Johansson. Early analog computers in Sweden - with examples from Chalmers University of Technology and the Swedish aerospace industry. *IEEE Annals of the History of Computing*, 18(4):27–33, 1996.

[JZ13] Nick D. James and Jeffery I. Zucker. A class of contracting stream operators. *The Computer Journal*, 56:15–33, 2013.

[Kle55] Stephen Kleene. Arithmetical predicates and function quantifiers. *Transactions of the American Mathematical Society*, 79:312–340, 1955.

[Lac55a] D. Lacombe. Extension de la notion de fonction récursive aux fonctions d'une ou plusieurs variables réelles i. *Comptes Rendus des Séances d l'Académie des Sciences, Paris*, 240:2478–2480, 1955.

[Lac55b] D. Lacombe. Extension de la notion de fonction récursive aux fonctions d'une ou plusieurs variables réelles ii. *Comptes Rendus des Séances d l'Académie des Sciences, Paris*, 241:13–14, 1955.

[Lac55c] D. Lacombe. Extension de la notion de fonction récursive aux fonctions d'une ou plusieurs variables réelles iii. *Comptes Rendus des Séances d l'Académie des Sciences, Paris*, 241:151–153, 1955.

[LR87] Leonard Lipshitz and Lee Rubel. A differentially algebraic replacement theorem. *Proceedings of the American Mathematical Society*, 99(2):367–372, 1987.

[MC04] Jerzy Mycka and José Félix Costa. Real recursive functions and their hierarchy. *Journal of Complexity*, 20(6):835–857, 2004.

[Mil08] Jonathan W. Mills. The nature of the extended analog computer. *Physica D Nonlinear Phenomena*, 237(9):1235–1256, 2008.

[Moo96] Cris Moore. Recursion theory on the reals and continuous-time computation. *Theoretical Computer Science*, 162(1):23–44, 1996.

[OLBC10] Frank W. J. Olver, Daniel W. Lozier, Ronald F. Boisvert, and Charles W. Clark. *NIST Handbook of Mathematical Functions*. Cambridge University Press, 2010.

[PE74] Marian Pour-El. Abstract computability and its relations to the general purpose analog computer. *Transactions of the American Mathematical Society*, 199:1–28, 1974.

[PER79] Marian Pour-El and Ian Richards. A computable ordinary differential equation which possesses no computable solution. *Annals of Mathematical Logic*, 17:61–90, 1979.

[Pou15] Amaury Pouly. *Continuous models of computation: from computability to complexity*. PhD thesis, École Polytechnique and Universidade do Algarve, 2015.

[RR06] Michael Renardy and Robert C. Rogers. *An introduction to partial differential equations*, volume 13. Springer-Verlag New York, second edition, 2006.

[Rub93] Lee A. Rubel. The extended analog computer. *Advances in Applied Mathematics*, 14(1):39–50, 1993.

[Sha41] Claude Shannon. Mathematical theory of the differential analyser. *Journal Mathematical Physics*, 20:337–354, 1941.

[Sma93] James S. Small. General-purpose electronic analog computing: 1945-1965. *IEEE Annals of the History of Computing*, 15(2):8–18, 1993.

[TT80] William Thompson and Peter G. Tait. *Treatise on Natural Philosophy*, volume 1. Cambridge University Press, 2nd edition, 1880. Part I.

[TZ00] John V. Tucker and Jeffery I. Zucker. Computable functions and semicomputable sets on many sorted algebras. In Samson Abramsky, Dov Gabbay, and Tom Maibaum, editors, *Handbook of Logic for Computer Science*, volume V of *University Series in Mathematics*, pages 317–523. Oxford University Press, 2000.

[TZ07] John V. Tucker and Jeffery I. Zucker. Computability of analog networks. *Theoretical Computer Science*, 371:115–146, 2007.

[TZ11] John V. Tucker and Jeffery I. Zucker. Continuity of operators on continuous and discrete time streams. *Theoretical Computer Science*, 412:3378–3403, 2011.

[TZ14] John V. Tucker and Jeffery I. Zucker. Computability of operators on continuous and discrete time streams. *Computability*, 3:9–44, 2014.

[Wei00] Klaus Weihrauch. *Computable Analysis — An Introduction.* Texts in Theoretical Computer Science. Springer-Verlag Berlin Heidelberg, 2000.