Computability with continuous-time dynamical systems

Daniel S. Graça[1,2]

[1]DM/FCT, Universidade do Algarve, Portugal
[2]SQIG, Instituto de Telecomunicações, Portugal

January 12, 2009


Outline

Motivation

There is a longstanding tradition of considering both discrete and continuous models for natural phenomena

- Leucippus and Democritus (ca. 400 BC) advocated that matter is not infinitely divisible and is constituted by particles, the *atoms*.
- In classical physics, time and space are idealized as continuous
- According to the work of Planck (ca. 1900), there is a minimum amount of time and space that can be measured.

While it is unclear whether real numbers have a physical meaning, they are mathematically useful for a large number of domains.

The digital paradigm

Church-Turing Thesis:

Every function computable according to the intuitive notion of algorithm is computable by a Turing machine

Not to confound with the following stronger claim:

Whatever can be calculated by a reasonable physical device is Turing computable.

So, what is the situation for analog computers?

Characteristics of analog computation

Usually agreed views in analog computation [Siegelmann]

- Any physical system or dynamical behavior in nature can be perceived as performing a computational process
- These processes can be modeled by dynamical systems, associated to some *state space*
- An analog computer should use a continuous state space

However, there is no widely accepted Universal Analog Computer, i.e., there is no counterpart of the Church-Turing Thesis for analog computation

Importance of analog computation

Many engineering problems are stated using real data/continuous dynamical systems.

- Digital computers truncate data, while analog computers do not. Can this be used as a future complement/alternative to digital microprocessors, where an analog "co-processor" would solve more efficiently certain classes of problems? (work of Jonathan Mills at University of Indiana)
- Can we relate problems over continuous dynamical systems with standard computability? (useful for control theory, etc.)
- Is there a Church-Turing Thesis for analog computation, or do we have to stick to a plethora of different models?

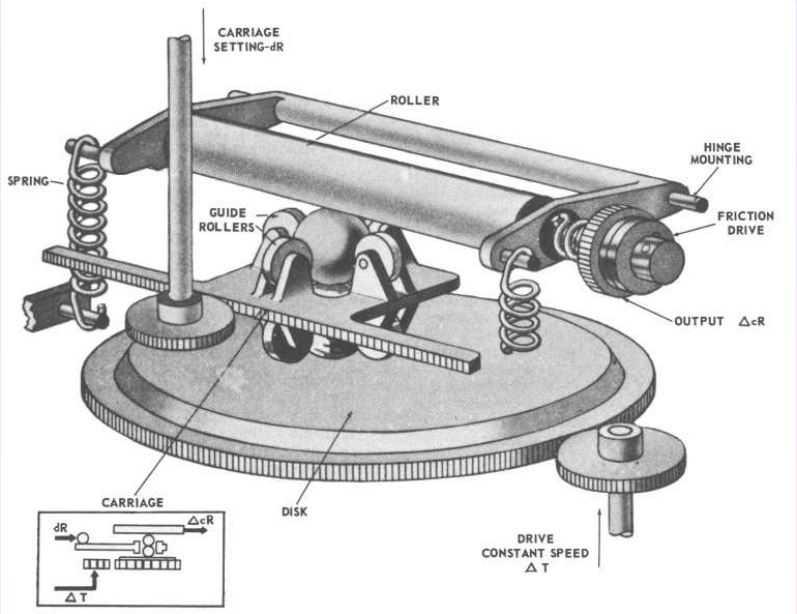
Motivations for our research

- Many models of analog computation found in the literature are based on characteristics borrowed from digital computers (e.g. treat real numbers as sequences of bits. Use dynamics that are not smooth), and may not be adequate for some applications
- Considering that classical models in Physics usually depend on analytic dynamics, it would be interesting to have an analog model of computation suited to this case
- By this reason, we focused our research on an older model of analog computation. With this we hope to gain some insight in the possibilities offered by analog computation

The differential analyzer

- The principles underlying this computing device were first described by Lord Kelvin in 1876
- A fully operational Differential Analyzer was assembled at MIT in 1931, under the supervision of V. Bush
- Several mechanical Differential Analyzers were built in the 1930s and 1940s, especially during the U.S. war effort. Their applications ranged from gunfire control up to aircraft design
- In the late forties, they were substituted by electronic versions that remained in use up to the beginnings of the 1970s

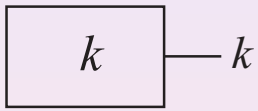
A mechanical integrator:



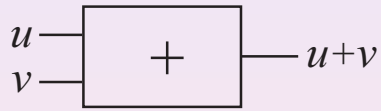
Bureau of Naval Personnel, *Basic Machines and How They Work*, 1964

The GPAC

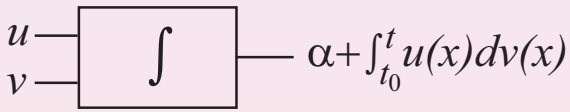
In 1941, Claude Shannon presented a paper entitled "Mathematical theory of the Differential Analyzer", where he first described the *General Purpose Analog Computer* (GPAC).



A constant unit associated to the real value k



An adder unit



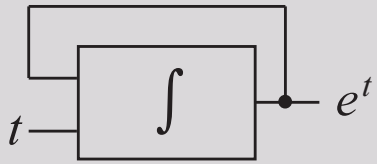
An integrator unit

A multiplier unit

Example

Example

Compute $y(x) = e^x$ with a GPAC



$$\begin{cases} y' = y \\ y(0) = 1 \end{cases}$$

Main features of the GPAC

- Real numbers are not treated as strings of digits
- Assumes continuous-time dynamics
- The computation is performed in *real time*: for a GPAC computing a function f, if an input x is given at time t, the output at time t is $f(x)$, i.e. the computation took 0 time units to be carried out.
- Generates analytic functions (i.e. it has smooth dynamics)

Shannon's algebraic-differential characterization

Definition

The unary function $y : \mathbb{R} \to \mathbb{R}$ is differentially algebraic (d.a.) on $I \subseteq \mathbb{R}$ if there is a nonzero polynomial p with real coefficients such that

$$p\left(x, y, y', ..., y^{(n)}\right) = 0, \quad \text{on } I. \tag{1}$$

Theorem (Shannon)

A unary function y is generated by a GPAC on an interval $I \subseteq \mathbb{R}$ iff it is d.a. on I.

This result implies that:

1. The GPAC computes polynomials, the exponential function, the usual trigonometric functions, their inverses as well as the finite composition and quotients of all these functions wherever they are well defined;
2. The Gamma function

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

and Riemann's zeta function

$$\zeta(x) = \sum_{n=1}^{\infty} \frac{1}{n^x}$$

cannot be computed by a GPAC.

Problems of Shannon's GPAC

- Shannon's assumes that his circuits have an output and that this output is unique. However, he does not provide any criteria to know when this happens (in particular, is this problem decidable?)
- The result relating the GPAC with d.a. functions has a gap (first noticed by Pour-El in 1974)

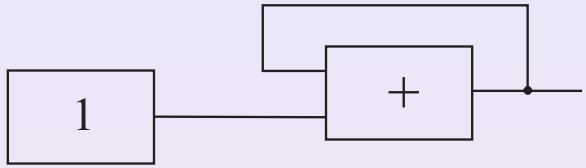


Figure 1: A circuit that admits no solutions as outputs.

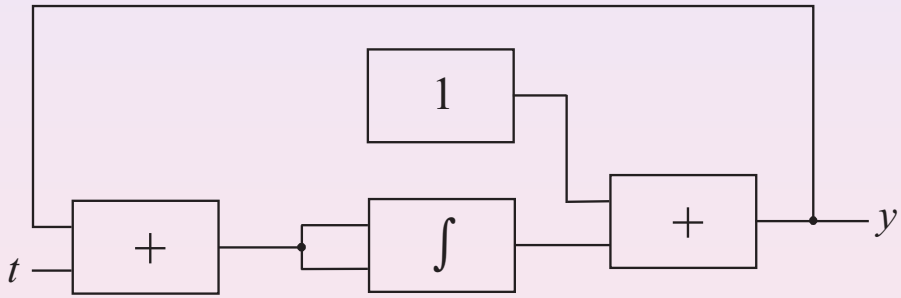


Figure 2: A circuit that admits two distinct solutions as outputs.

Other work

- In 1974, M. B. Pour-El pointed out a gap on Shannon's proof. She also redefined the GPAC in order to prove Theorem 1. Nevertheless, it is not clear if her model, based on system of differential equations, has anything to do with the Differential Analyzer.
- Further refinements where made by Lipshitz and Rubel (1987).

The Extended Analog Computer

In 1993, L.A. Rubel presented an extension of the GPAC, the Extended Analog Computer (EAC). This model has the following features:

1. It has all the units of the GPAC, plus units that can compute Partial Differential Equations (PDEs), inverse functions, and limits (this later, with some restrictions);
2. It can compute functions that are not computable by the GPAC (e.g. Γ, ζ) and solve problems not solvable by a GPAC (namely problems involving PDEs and limits);
3. Contrarily to the GPAC, it is not based on any physical device, and its realizability remains unknown (**remark:** recently, researchers from the Indiana University were able to produce analog VLSI units capable of computing PDEs)

Our approach

Use Shannon's model, but with restrictions on the layout of the circuits

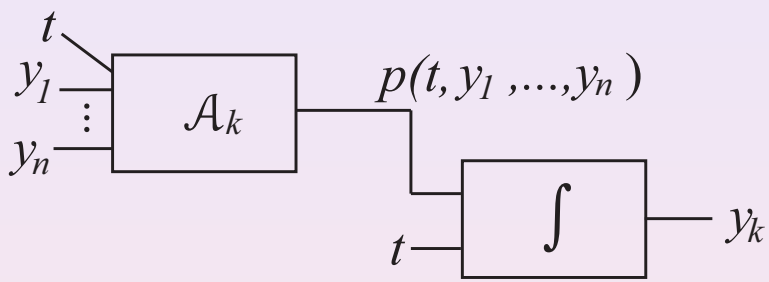
1. Polynomial circuits: acyclic circuits built in layers, without using integrators.

Lemma

Polynomial circuits generate exactly the class of polynomial functions

2. Use polynomial circuit as building blocks to obtain polynomial GPACs.

The basic block of the PGPAC



Here t is the input and y_k is the output of the kth integrator. $\mathcal{A}_k$ is a polynomial circuit.

Main results I (Graça, Costa)

Theorem (Shannon $\Longrightarrow$ part)

If a unary function y is generated by a PGPAC in $I \subseteq \mathbb{R}$ then it is d.a. on I.

Theorem (Shannon $\Longleftarrow$ part)

If a unary function y is d.a. in $I \subseteq \mathbb{R}$ then it is generated by a PGPAC in some $I' \subseteq I$, where I' has non-empty interior.

Corollary (connection with dynamical systems)

y is generated by a PGPAC iff it is a solution of $\mathbf{y}' = \mathbf{p}(\mathbf{y}, x)$, where $\mathbf{p}$ is a vector of polynomials.

- In particular, the last result guarantees that the PGPAC is well behaved.
- Because our model is well behaved, is based on circuits (Shannon's approach), can be written as a system of differential equations (Pour-El's approach), and has the main properties stated by Shannon and Pour-El, solving previous problems, we decided to call it "the GPAC".
- However, we drop the initial motivation based in circuits. Instead we stress on the fact that the GPAC generate solutions of polynomial systems of the form $\mathbf{y}' = \mathbf{p}(\mathbf{y}, x)$. We call solutions of these systems PIVP functions. We think this a more elegant and useful characterization of the GPAC.

Properties of PIVP functions

The PIVP functions are closed under the following operations (as far as we know, these properties have only been reported in the literature for the broader case of differentially algebraic functions):

- Field operations $+, -, \times, /$
- Composition
- Differentiation
- Compositional inverses

Corollary

All closed-form functions (i.e. elementary functions in Analysis which, informally, correspond to the functions obtained from the rational functions, sin, cos, exp through finitely many compositions and inversions) are PIVP functions.

Main results II (Graça, Campagnolo, Buescu)

Theorem (robust continuous-time simulation of TM's)

Let $\theta : \mathbb{N}^3 \to \mathbb{N}^3$ be the encoding of the transition function of a Turing machine M and let $0 < \delta < \varepsilon < 1/2$. Then there is a PIVP function $g_M : \mathbb{R}^6 \to \mathbb{R}^6$ such that the ODE $z' = g_M(z,t)$ robustly simulates M in the following sense: there is some $0 < \eta < 1/2$ such that for all g satisfying $\|g - g_M\|_\infty < 1/2$, and for all $\bar{x}_0 \in \mathbb{R}^3$ satisfying $\|\bar{x}_0 - x_0\|_\infty \leq \varepsilon$, where $x_0 \in \mathbb{N}^3$ represents an initial configuration, the solution z of

$$z' = g(z,t), \qquad z(0) = (\bar{x}_0, \bar{x}_0)$$

has the following property: for all $j \in \mathbb{N}$ and for all $t \in [j, j+1/2]$,

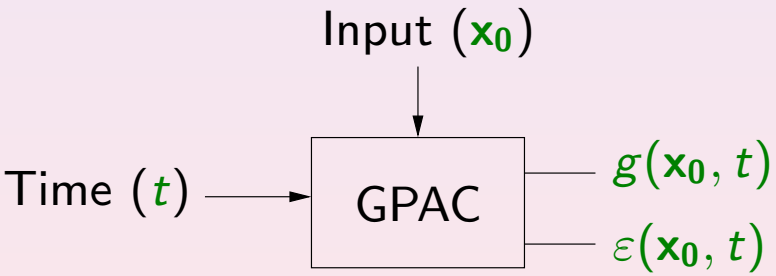
$$\left\| z_2(t) - \theta^{[j]}(x_0) \right\|_\infty \leq \eta.$$

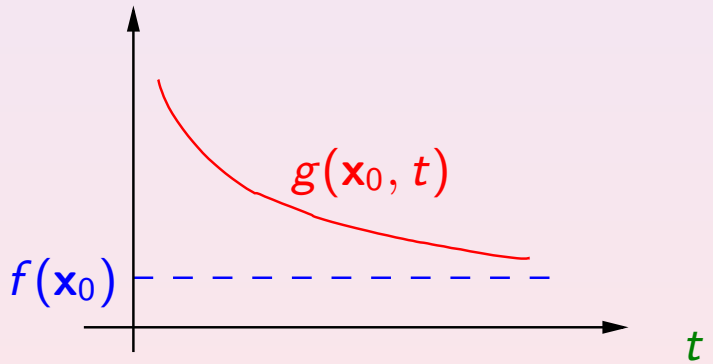
Computing Γ with a GPAC

Problem: Γ is computable according to Computable Analysis, but not computable by the GPAC.

Solution: Change the notion of computability associated to a GPAC.

The idea: Apply a limit procedure, similarly to the Computable Analysis case.





This can be achieved as follows:

1. use initial settings on integrators to represent the initial input $x \in \mathbb{R}^n$
2. use the usual input as a time variable t
3. then $f : \mathbb{R}^n \to \mathbb{R}$ is GPAC*-computable if there is a GPAC with two outputs $g(x, t)$ and $\varepsilon(x, t)$ satisfying:
   1. $\lim_{t\to\infty} \|\varepsilon(x, t)\| = 0$;
   2. $\|g(x, t) - f(x)\| \leq \varepsilon(x, t)$

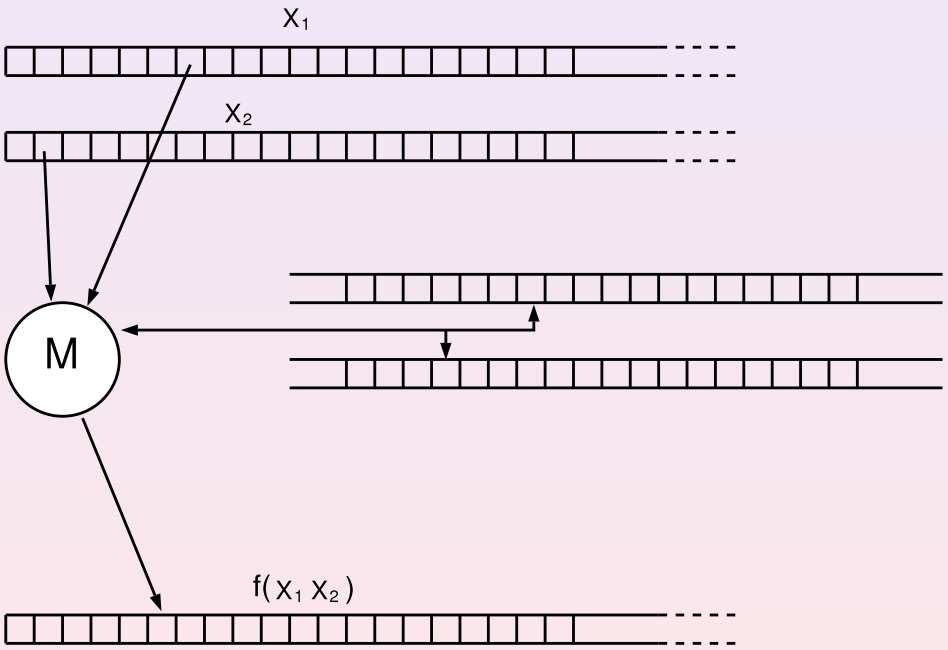
Theorem

Both Γ and ζ are GPAC-computable*

Computable analysis: Type-2 machines

Coding a real number x

For a sequence $(y_n, z_n)_{n\in\mathbb{N}}$, where $y_i, z_i \in \mathbb{N}$, we write $(y_n, z_n) \rightsquigarrow x$ iff $\forall i, |x - (-1)^{y_i} \frac{z_i}{2^i}| \leq \frac{1}{2^i}$



M behaves like a Turing Machine

An input x is coded on a read-only one-way input tape and the output $f(x)$ is coded on a write-only one-way output tape.

If there is a Type-2 machine with these properties, we say that $f : \mathbb{R} \to \mathbb{R}$ is computable.

Main results III (Bournez, Campagnolo, Graça, Hainry)

Theorem

Let $f : [a, b] \to \mathbb{R}$ be a real function. Then f is computable (Computable Analysis sense) iff it is GPAC-computable.

Conclusions/perspectives

- We have presented a model that relates in a very natural way to a large class of dynamical systems
- This model robustly simulates Turing machines (achieves Type-1 computability)
- It is equivalent to computable analysis on compact intervals
- What happens for unbounded domains?
- Is this a suitable candidate as a reference model for an analog "Church-Turing" thesis?

Selected references

- [Sha41] C. E. Shannon. Mathematical theory of the differential analyzer. J. Math. Phys. MIT, 20:337–354, 1941.
- [Pou74] M. B. Pour-El. Abstract computability and its relations to the general purpose analog computer. Trans. Amer. Math. Soc., 199:1–28, 1974
- [GC03] D. S. Graça and J. F. Costa. Analog computers and recursive functions over the reals. J. Complexity, 19(5):644–664, 2003
- [DCGH07] O. Bournez, M. L. Campagnolo, D. S. Graça, and E. Hainry. Polynomial differential equations compute all real computable functions on computable compact intervals. J. Complexity, 23(3):317–335, 2007
- [GCB08] D. S. Graça, M. L. Campagnolo, and J. Buescu. Computability with polynomial differential equations. Adv. Appl. Math., 40(3):330–349, 2008

Thank you!