Course - The General Purpose Analog Computers. Differential Analyzers

Olivier Bournez

November 9, 2018

In 1941, Claude Shannon introduced in [Sha41] the GPAC model as a model for the Differential Analyzer [Bus31], on which he worked as an operator.

1 Differential Analysers

Differential Analysers are mechanical (and later on electronics) continuous time analog machines.

The Differential Analyzer was used from the 1930s to the early 60s to solve numerical problems. For example, differential equations were used to solve ballistics problems. These devices were first built with mechanical components and later evolved to electronic versions.

One of the MIT Differential Analyser

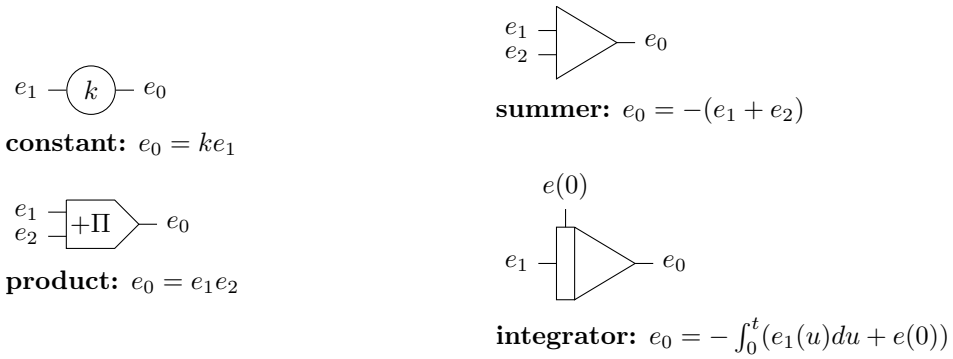


Figure 1: Circuit presentation of the GPAC: a circuit built from basic units. Presentation of the 4 types of units: constant, adder, multiplier, and integrator.

First differential analyzers were mechanical. Electronic versions were used from late 40s until 70s.

Analog paradigm is selling some modern differential analyzers. I have one *Analog Paradigm Model-1* in my office at Ecole Polytechnique.

Underlying principles of the Differential Analysers can be attributed to Lord Kelvin 1876. First ever built machine was built under the supervision of V. Bush 1931 at MIT. Applications were from gunfire control up to aircraft design. They were intensively used during U.S. war effort.

2 The GPAC model

GPAC stands for *General Purpose Analog Computer*.

The GPAC was originally introduced by Shannon in [Sha41], and further refined in [PE74, LR87, GC03, Gra04].

Basically, a GPAC is any circuit that can be build from the 4 basic units of Figure 1, that is to say from basic units realizing constants, additions, multiplications and integrations, all of them working over analog real quantities (that were corresponding to angles in the mechanical Differential Analysers, and later on to voltage in the electronic versions).

Actually, not all kinds of interconnections must be allowed since this may lead to undesirable behavior (*e.g.* non-unique outputs. For further details, refer to [GC03]).

In other words, a GPAC may be seen as a circuit built of interconnected black boxes, whose behavior is given by Figure 1, where inputs are functions of an independent variable called the *time* (in an electronic Differential Analyzer, inputs usually correspond to electronic voltages). These black boxes add or multiply two inputs, generate a constant, or solve a particular kind of Initial Value Problem defined with an Ordinary Differential Equation (ODE for short).

Figures 3 illustrates for example how the sine function can generated using

two integrators, with suitable initial state, as being the solution of ordinary differential equation

$$\begin{cases} y'(t) = z(t) \\ z'(t) = -y(t) \end{cases}$$

with suitable initial conditions.

The original GPAC model introduced by Shannon has the feature that it works in *real time*: for example if the input t is updated in the GPAC circuit of Figure 3, then the output is immediately updated for the corresponding value of t.

Shannon, in his original paper, already mentioned that the GPAC generates polynomials, the exponential function, the usual trigonometric functions, their inverses, and their composition. More generally, Shannon claimed that all functions generated by a GPAC are differentially algebraic in the sense of the following definition.

Definition 1 *A unary function y is differentially algebraic (d.a.) on the interval I if there exists an $n \in \mathbb{N}$ and a nonzero polynomial p with real coefficients such that*

$$p\left(t, y, y', ..., y^{(n)}\right) = 0, \qquad \text{on } I. \tag{1}$$

As a corollary, and noting that the Gamma function $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$ is not d.a. [Rub89], we get that

Proposition 1 *The Gamma function cannot be generated by a GPAC.*

Another famous example of not d.a. function is Riemann's Zeta function $\zeta(x) = \sum_{k=0}^{\infty} \frac{1}{k^x}$ (proof of non d.a. by Hilbert).

If we have in mind that these functions are known to be computable under the computable analysis framework [PER89], the previous result has long been interpreted as evidence that the GPAC is a somewhat weaker model than computable analysis.

However, Shannon's proof relating functions generated by GPACs with d.a. functions was incomplete (as pointed out and partially corrected in [PE74, LR87]). Actually, as pointed out in [GC03], the original GPAC model suffers from several robustness problems.

3 GPAC and polynomomial Initial Value Problems

However, for the more robust class of GPACs defined in [GC03] by restricting the possible layout of a GPAC, the following stronger property holds:

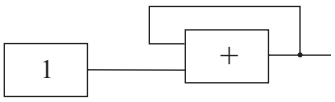


Figure 1: A circuit that admits no solutions as outputs.

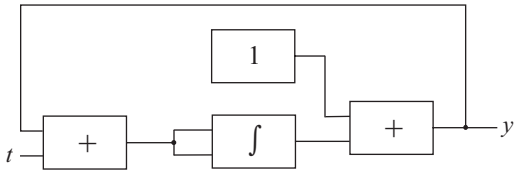


Figure 2: A circuit that admits two distinct solutions as outputs.

Figure 2: Problematic circuits: (I apologize: the representation of basic blocks differ from other figures as the current images are taken from some other source)
.

Proposition 2 *A scalar function $f : \mathbb{R} \to \mathbb{R}$ is generated by a GPAC iff it is a component of the solution of a system*

$$y' = p(y, t), \tag{2}$$

where p is a vector of polynomials. A function $f : \mathbb{R} \to \mathbb{R}^k$ is generated by a GPAC iff all of its components are.

Basically, the idea of the proof is just to introduce a variable for each output of a basic unit, and write the corresponding ordinary differential equation (ODE), and observe that it can be written as an ODE with a polynomial right hand side.

For a concrete example of Proposition 2, see Figure 3. From now on, we will mostly talk about GPACs as being systems of ODEs of the type (2).

We say that a function $f : \mathbb{R} \to \mathbb{R}$ is generable (by a GPAC) if and only if it corresponds to some component of a solution of such a polynomial initial value problem (2).

The discussion on how to go from univariate to multivariate functions, that is to say from functions $f : \mathbb{R} \to \mathbb{R}^m$ to functions $f : \mathbb{R}^n \to \mathbb{R}^m$ is briefly discussed in [Sha41], but no clear definitions and results for this case have been stated or proved previously, up to our knowledge. This is the purpose of the following part of the course. Another objective is to introduce basic measures of the resources used by a GPAC (in particular on the growth of functions), which might be used in the future to establish complexity results for functions generated with GPACs.

We will introduce in another course formally the notion of *generable* functions which are solutions of a polynomial initial-value problem (PIVP), and

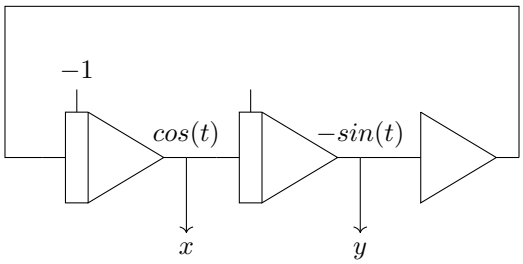


Figure 3: Example of GPAC circuit: computing sine and cosine with two variables

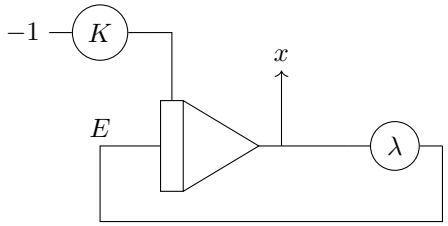
generalize this notion to several input variables. We will prove that this class enjoys a number of stability and robustness properties.

4 Programming with the GPAC

We here provide some examples of programs.

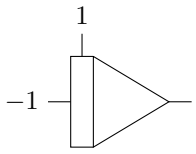
4.1 Exponential

Exponential: $E(t) = Kexp(-\lambda t)$

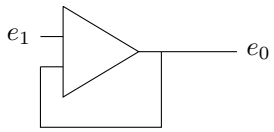


4.2 Linear operations

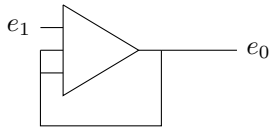
Linear function: $x(t) = t$



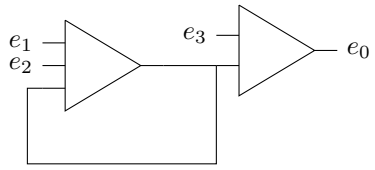
Linear operations:



$$e_0 = -\frac{e_1}{2}$$



$$e_0 = -\frac{e_1}{3}$$

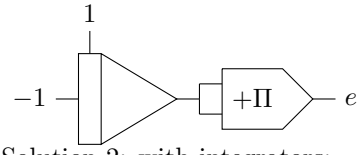


$$e_0 = \frac{e_1 + e_2}{2} - e_3$$

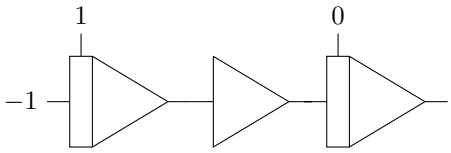
4.3 Polynomials

Parabola: $x(t) = (-1 + t)^2$

Solution 1: With a product:

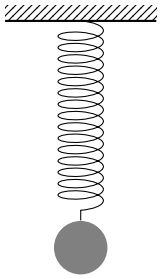


Solution 2: with integrators:



4.4 Damped Spring

Damped spring



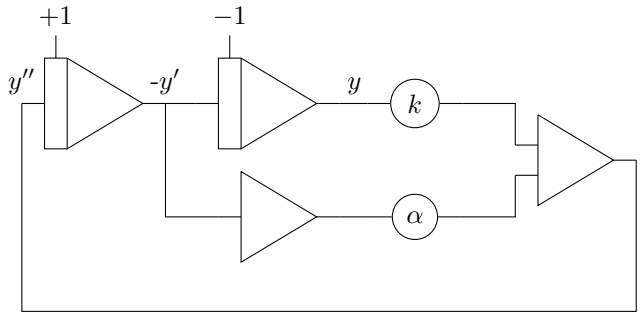
spring constant k

$$my'' + \alpha y' + ky = 0$$

Hence

$$y'' = -\frac{\alpha y' + ky}{m}$$

Damped spring: $y'' = -\alpha y' + ky$:
$m = 1$.

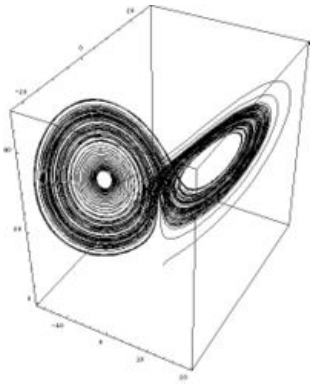


Example:

- $k = 0.8$
- $\alpha = 0.2$

4.5 Lorenz's attractor

Lorenz's attractor:



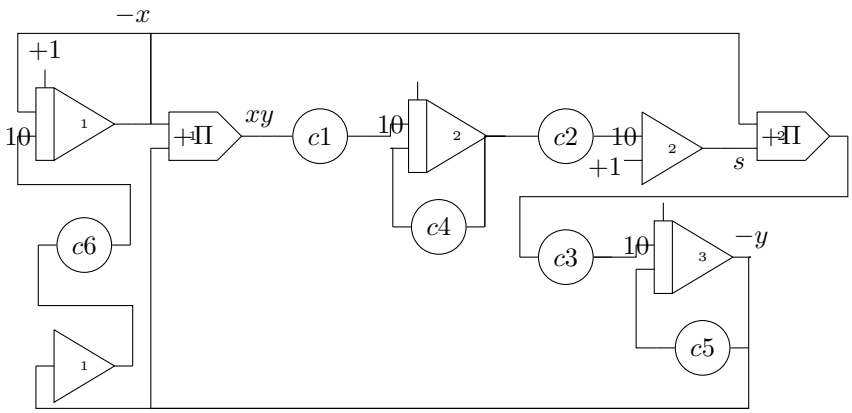
$$\begin{cases} x' &= \sigma(y-x) \\ y' &= \rho x - y - xz \\ z' &= xz - \beta z \end{cases}$$

$\sigma = 10, \beta = 8/3, \rho = 28$

Rescaling:

$$\begin{cases} x &= \int 1.8y - xdt + C \\ s &= 1 - 2.678z \\ y &= \int 1.5556xs - 0.1ydt \\ z &= \int 1.5xy - 0.2667zdt \end{cases}$$

The program:



$c1 = 0.15$, $c2 = 0.268$, $c3 = 0.1536$, $c4 = 0.2667$, $c5 = 0.1$, $c6 = 0.18$

References

[Bus31] V. Bush. The differential analyzer. A new machine for solving differential equations. *J. Franklin Inst.*, 212:447–488, 1931.

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

[Gra04] Daniel S. Graça. Some recent developments on Shannon's General Purpose Analog Computer. *Mathematical Logic Quaterly*, 50(4-5):473–485, 2004.

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

[PE74] M. B. Pour-El. Abstract computability and its relations to the general purpose analog computer. *Trans. Amer. Math. Soc.*, 199:1–28, 1974.

[PER89] M. B. Pour-El and J. I. Richards. *Computability in Analysis and Physics*. Springer, 1989.

[Rub89] L. A. Rubel. A survey of transcendentally transcendental functions. *American Mathematical Monthly*, 96(9):777–788, 1989.

[Sha41] C. E. Shannon. Mathematical theory of the differential analyser. *Journal of Mathematics and Physics MIT*, 20:337–354, 1941.