EFFICIENT ALGORITHMIC TOOLS FOR EXPERIMENTAL MATHEMATICS

POST-DOC PROPOSITION BY
BRUNO SALVY & GILLES VILLARD,
ARIC TEAM, ENS LYON

The experimental approach in mathematics consists of three steps: computation of high-precision approximations; computation of conjectured relations from these approximations; proofs of these conjectures. These three steps require efficient algorithms, of different natures. The aim of this post-doc is to advance the algorithmic knowledge on tools helping the discovery of conjectures. In this context, two main tools are used to discover relations between approximations: Padé-Hermite approximants, that uncover linear relations with polynomial coefficients between formal power series; the LLL algorithm for short vectors in Euclidean lattices, which is used to conjecture linear relations with integer coefficients between real numbers known with high precision. Popular and efficient implementations of these algorithms are developed in the AriC team, in the `gfun` Maple package [7] and in the `fplll` and `hplll` C++ libraries, partially accessible from Sage [8].

The aim of this post-doc is to study how these computations can be performed more efficiently when the input data possesses an extra structure. In both cases, providing efficient and easy-to-use implementations of the state of the art could be a good starting point for further exploration.

1. Structured Padé-Hermite approximants

Given formal power series $\mathbf{F} = (f_1, \dots, f_n)$ in $\mathbb{K}[[x]]^n$ where $\mathbb{K}$ is an arbitrary field, and given nonnegative integers $\mathbf{d} = (d_1, \dots, d_n)$, an n-tuple of polynomials $\mathbf{P} = (P_1, \dots, P_n)$ in $\mathbb{K}[x]^n$ is called a *Padé-Hermite approximant* of $\mathbf{F}$ of type $\mathbf{d}$ when

$$\mathbf{P} \cdot \mathbf{F} = P_1 f_1 + \cdots + P_n f_n = O(x^N) \qquad (1)$$

with $N = (d_1 + \cdots + d_n) + n - 1$ and $\deg P_i \le d_i$ for $1 \le i \le n$.

These approximants have been introduced by Hermite in his proof of the transcendence of e and remain an important tool in recent proofs of transcendence [1]. The coefficients of the polynomials P_i are solutions of a linear system and can therefore be computed in $O(N^3)$ operations in $\mathbb{K}$, or even $O(N^\theta)$ operations in $\mathbb{K}$, where $\theta \in [2, 3]$ is a feasible exponent for the complexity of matrix product.

The n-tuples $\mathbf{P}$ obeying the relation (1) form a free sub-module of $\mathbb{K}[x]^n$ of rank n. One can compute special bases of it, called minimal bases, or σ-bases, that contain a Padé-Hermite approximant of type d in only $O(nN^2)$ operations in $\mathbb{K}$, by an algorithm that generalizes the extended gcd algorithm. It is even possible to go further and it has been proved that an algorithm due to Beckermann and Labahn in 1994 [2] performs this computation in only $O(n^{\theta-1}N\log^2 N)$ operations in $\mathbb{K}$, where again θ is the exponent of the complexity of matrix product. The same complexity can also be achieved by a probabilistic algorithm exploiting the structure of the matrix underlying the linear algebra system [3], which is of quasi-Toeplitz type (a generalization of Sylvester's matrix underlying the gcd). Generalizations to other types of structures, still in the same complexity, are also possible [5]. With respect to the sizes of the input and output of the Padé-Hermite problem, that are both in $O(nN)$, the complexities of these algorithms are close to optimal.

In large size computations arising in experimental mathematics and notably for applications in combinatorics or physics, the input $\mathbf{F}$ itself is structured and can be stored in size only N. Two important special cases are *differential approximants*, where $\mathbf{F} = (f, f', f'', \dots, f^{(n-1)})$ is built from the first derivatives of a given power series f; and *algebraic approximants*, where it is built from the first powers of the power series. The aim is to conjecture linear differential or polynomial equations annihilating the power series. Besides, the real aim is not a full basis of the module of approximants, but a single minimal approximant. The input and the output then have size only N. A natural and important question is therefore

> *Can one exploit this supplementary structure of the quasi-Toeplitz matrix to decrease the complexity of the computation with respect to N?*

2. Sparse Vectors in Lattices

Analogous questions arise in Euclidean lattices, that are modules over the ring of integers rather than polynomials. The main algorithm in this area is LLL, invented by Lenstra, Lenstra and Lovasz in 1982 [4], where they gave the first algorithm for the factorization of polynomials in $\mathbb{Q}[z]$ that runs in polynomial complexity. Since then, an abundant literature has been devoted to applications, extensions and improvements of this algorithm. A good survey of the recent state of the art has been published after a conference organized on the occasion of its 25 years [6]. The chapters due to Damien Stehlé and Guillaume Hanrot (both members of AriC) are the most relevant to the applications to experimental mathematics. The LLL algorithm discovers short vectors in a Euclidean lattice — a discrete subgroup of $\mathbb{R}^d$ — in the norm L^2 sense. Finding the non-zero vectors of minimal norm is called the *shortest vector problem* and known to be NP-hard. In polynomial time, LLL manages to find non-zero vectors whose norms are not necessarily the shortest possible ones, but that are at a controlled factor of them. Using the known bounds on this factor makes it possible to encode approximation problem into lattices in such a way that the vectors returned by LLL are indeed minimal. The main use of this technique in experimental mathematics is the quest for linear dependence over $\mathbb{Q}$ between real numbers given by high precision approximations. The simplest example is given by the first powers $1, \alpha, \alpha^2, \dots, \alpha^d$ of a real number α, where a linear relation yields a conjecture of the algebraicity of α with a candidate minimal polynomial. In Maple, the function `identify` uses the numbers $(1, \sqrt{2}, \sqrt{3}, \pi, \ln 2, \ln 3, \zeta(3), \zeta(5), x)$, where x is given by the user, to "recognize" x if it is a linear combination of the preceding ones over $\mathbb{Q}$. Nonetheless, the complexity of LLL, even though it is polynomial, remains relatively large and a straightforward extension of this functionality to bases of thousands of constants is impossible.

Here again, this specific question offers an underlying structure that is not exploited by the algorithm: all the constants except one are known in advance and do not change from one execution of the algorithm to the next; rather than being short, the vectors that are desired should more importantly be sparse. The question to be explored is thus:

> *Can one exploit the supplementary structure of the problem of identifying constants so as to reduce the complexity of the computation with respect to dimension?*

References

[1] Boris Adamczewski and Tanguy Rivoal. Exceptional values of E-functions at algebraic points. Technical report, arXiv, August 2017.

[2] Bernhard Beckermann and George Labahn. A uniform approach for the fast computation of matrix-type Padé approximants. *SIAM Journal on Matrix Analysis and Applications*, 15(3):804–823, July 1994.

[3] Alin Bostan, Claude-Pierre Jeannerod, and Éric Schost. Solving structured linear systems with large displacement rank. *Theoretical Computer Science*, 407(1-3):155–181, 2008.

[4] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982. 10.1007/BF01457454.

[5] Vincent Neiger. *Bases of relations in one or several variables: fast algorithms and applications.* Thèse de doctorat, École Normale Supérieure de Lyon - University of Waterloo, November 2016.

[6] Phong Q. Nguyen and Brigitte Vallée, editors. *The LLL algorithm.* Information Security and Cryptography. Springer-Verlag, Berlin, 2010. Survey and applications.

[7] Bruno Salvy and Paul Zimmermann. Gfun: a Maple package for the manipulation of generating and holonomic functions in one variable. *ACM Transactions on Mathematical Software*, 20(2):163–177, 1994.

[8] The FPLLL development team. fplll, a lattice reduction library. Available at `https://github.com/fplll/fplll`, 2016.