Algebraic and Arithmetic Attributes of Hypergeometric Functions in SageMath

Xavier Caruso
CNRS; IMB, Université de Bordeaux
Talence, France
xavier@caruso.ovh

Florian Fürnsinn
University of Vienna, Faculty of Mathematics
Vienna, Austria
florian.fuernsinn@univie.ac.at

Abstract

We report on implementations for algorithms treating algebraic and arithmetic properties of hypergeometric functions in the computer algebra system SageMath. We treat hypergeometric series over the rational numbers, over finite fields, and over the p-adics. Among other things, we provide implementations deciding algebraicity, computing valuations, and computing minimal polynomials in positive characteristic.

1 Introduction

A *hypergeometric function* with rational *top parameters* $\boldsymbol{\alpha} \coloneqq (\alpha_1, \dots, \alpha_n) \in \mathbb{Q}^n$ and *bottom parameters* $\boldsymbol{\beta} \coloneqq (\beta_1, \dots, \beta_m) \in \mathbb{Q}^m$ is defined as the power series

$$ {}_nF_m(\boldsymbol{\alpha}, \boldsymbol{\beta}; x) \coloneqq \sum_{k=0}^{\infty} \frac{(\alpha_1)_k \cdots (\alpha_n)_k}{(\beta_1)_k \cdots (\beta_m)_k} \cdot \frac{x^k}{k!} \in \mathbb{Q}[[x]], \qquad (1)$$

where $(\gamma)_k \coloneqq \gamma(\gamma+1)\cdots(\gamma+k-1)$ denotes the *rising factorial* or *Pochhammer symbol*.

The hypergeometric function ${}_nF_m(\boldsymbol{\alpha}, \boldsymbol{\beta}; x)$ is the solution of the differential equation $\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\beta}, x)y(x) = 0$ with, denoting $\vartheta = x\frac{\mathrm{d}}{\mathrm{d}x}$,

$$\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\beta}, x) \coloneqq \big(x(\vartheta+\alpha_1)\cdots(\vartheta+\alpha_n) - \vartheta(\vartheta-\beta_1)\cdots(\vartheta-\beta_m)\big).$$

A power series $f(x) \in K[[x]]$ is called *algebraic* if there exists a nonzero polynomial $P(x, y) \in K[x, y]$, such that $f(x, y(x)) = 0$. Similarly, a power series $f(x) \in \mathbb{Q}[[x]]$ is called *D-finite*, if there exists a nonzero differential operator $L \in \mathbb{Q}[x]\langle\partial\rangle$, such that $Lf(x) = 0$. Moreover, it is called *globally bounded*, if it has a a positive radius of convergence and there exist two nonzero integers $\alpha, \beta \in \mathbb{Z}$, such that $\beta f(\alpha x) \in \mathbb{Z}[[x]]$.

Hypergeometric functions are clearly D-finite, and the sets of parameters for which they are algebraic or globally bounded are fully classified. Moreover, their reductions modulo p in $\mathbb{F}_p[[x]]$ are algebraic, whenever they are defined. These properties make them ideal test cases for conjectures about algebraic and arithmetic properties of D-finite series. The need to use a software for formulating and checking such conjectures appeared to the authors when they worked, jointly with Vargas-Montoya, on Galois groups of reductions modulo primes of D-finite series [5].

The present article is the outcome of this observation. It reports on an open source SageMath package designed to manipulate hypergeometric functions in various situations: over the rationals, over finite fields and over p-adic fields. To provide these functionalities it was necessary to develop new algorithms. They are presented mostly in [4].

At the time being, our package is submitted for integration in a future release of SageMath [3]. In this paper, we shortly showcase the main features of our package. Our presentation is complemented by an interactive worksheet gathering all the examples presented below, available at:
https://xavier.caruso.ovh/notebook/hypergeometric-functions/
This interactive worksheet supports editing the examples and exploring new cases.

For the complete documentation of our package, we refer to the Section *Hypergeometric functions over arbitrary rings* in the reference manual of SageMath, available online here.

2 Setup

Hypergeometric functions are implemented in SageMath as elements of a symbolic ring.

```
 In:   f = hypergeometric([1/3, 2/3], [1/2], x)
 In:   f
Out:   hypergeometric((1/3, 2/3), (1/2,), x)
 In:   f.parent()
Out:   Symbolic Ring
```

We propose an implementation of algebraic and arithmetic properties of them, accessed by creating hypergeometric functions where the argument is an element of (the fraction field of) a polynomial ring or a power series ring.

```
 In:   S.<x> = QQ[]
 In:   f = hypergeometric([1/3, 2/3], [1/2], x)
 In:   f
Out:   hypergeometric((1/3, 2/3), (1/2,), x)
 In:   f.parent()
Out:   Hypergeometric functions in x over Rational Field
```

We emphasize that our package allows for quite general parameters, even permitting to have nonnegative integers as bottom parameters as soon as they are compensated by an appropriate top parameter. Compare:

```
 In:   hypergeometric([-1], [-2], x)
Out:   hypergeometric((-1,), (-2,), x)
 In:   hypergeometric([-2], [-1], x)
Out:   ValueError: the parameters ((-2,), (-1,))
        do not define a hypergeometric function
```

The main functions we will use throughout the following demonstration are $f(x)$, Christol's example of a hypergeometric function that is not known to be a diagonal; $g(x)$, where $g(16x^2)$ is the generating function of Gessel excursions, and the somewhat obscure function $h(x)$ that illustrates many phenomena.

The second named author was funded by a DOC Fellowship (27150) of the Austrian Academy of Sciences at the University of Vienna. Further he thanks the French–Austrian project EAGLES (ANR-22-CE91-0007 & FWF grant 10.55776/I6130) for financial support.
The authors thank Austria's Agency for Education and Internationalisation (OeAD) and Campus France for providing funding for research stays via WTZ collaboration project/Amadeus project FR02/2024.

```
In:  f = hypergeometric([1/9, 4/9, 5/9], [1/3, 1], x)
In:  g = hypergeometric([1/2, 5/6, 1], [5/3, 2], x)
In:  h = hypergeometric([1/5, 1/5, 1/5, 1/5],
      [1/3, 3^10/5 - 1], x)
```

3 Hypergeometric functions over $\mathbb{Q}$

Global Boundedness. Globally bounded hypergeometric functions have been fully classified by Christol [7]. We implement his criterion in the method `is_globally_bounded()`.

```
 In:  f.is_globally_bounded()
Out:  True
 In:  h.is_globally_bounded()
Out:  False
```

Christol's elegant criterion, essentially depends on the relative positions of the decimal parts $\{\Delta\alpha_i\}$ and $\{\Delta\beta_i\}$, for $\Delta \in \mathbb{Z}/d\mathbb{Z}$, where d is the least common denominator of the parameters.

Algebraicity. Similar in spirit, all algebraic hypergeometric functions have been classified. We implemented the corresponding criterion in the method `is_algebraic()`.

```
 In:  f.is_algebraic()
Out:  False
 In:  g.is_algebraic()
Out:  True
```

In case there are no integer differences between top and bottom parameters Christol [7], and Beukers and Heckman [1] provided the classification, a criterion, very similar to the classification of global boundedness, depending on relative positions of decimal parts. In case of integer differences, the criterion was extended by the second author and Yurkevich [8].

Primes With Good Reductions. We say that a series $s(x) \in \mathbb{Q}[[x]]$ has good reduction at p, if all its coefficients have denominator coprime with p. For large enough prime numbers, this property only depends on the congruence class of p modulo d. The method `good_reduction_primes()` computes this and outputs a set of prime numbers depending on congruence classes (implemented in SageMath by the first author [2]).

```
 In:  g.good_reduction_primes()
Out:  Set of all prime numbers with 2 excluded:
       3, 5, 7, 11, ...
 In:  h.good_reduction_primes()
Out:  Set of prime numbers congruent to 1, 8, 11 modulo 15
       with 17, 167, 677, 857, ..., 29327, 29387 included
       and 23, 83, 113, 173, ..., 58913, 58943 excluded:
       11, 17, 31, 41, ...
```

Similar to the techniques used for the classification of globally bounded hypergeometric functions, the authors showed in [5] and [4, § 3.1] that for large enough prime numbers it only depends on their congruence class modulo d, whether the series has nonnegative p-adic valuation, *i.e.*, can be reduced modulo p. Our implementation tests for small prime numbers p, whether the p-adic valuation is positive, using the method `valuation()`, showcased in Section 5. This is done for all primes smaller than the bound, after which regularity is ensured, and additionally for one prime number for each congruence class larger than this bound. We remark that alternative methods closely related to Christol's work were detailed in [5, § 3.1], which are not implemented.

4 Hypergeometric functions over $\mathbb{F}_p$

When a hypergeometric function $h(x)$ has good reduction at a prime p, we can form $h(x) \bmod p \in \mathbb{F}_p[[x]]$ and study its properties. It is known for example that the latter is always algebraic [6, 10, 9, 5, 4]. The main ingredients beyond this result are *section operators*, which allow to build what we call *Dwork relations* (a writing of hypergeometric function as a polynomial linear combination of p-th powers of other ones) and eventually to find annihilating polynomials. All these steps are implemented in our package.

Sections. For an integer r, we define the r-th section operator

$$\begin{array}{ccc} \mathbb{F}_p[[x]] & \longrightarrow & \mathbb{F}_p[[x]] \\ \sum_{k=0}^{\infty} a_k x^k & \mapsto & \sum_{k=0}^{\infty} a_{kp+r} x^k \end{array}$$

In [4, §3.2], it was shown that sections of hypergeometric functions can always be written as a product of a monomial and another hypergeometric function. Those can be computed using the method `section()`.

```
 In:  f19 = f % 19
 In:  f19.section(0)
Out:  hypergeometric((1/9, 4/9, 5/9), (1/3, 1), x)
 In:  f19.section(1)
Out:  14*hypergeometric((1/9, 4/9, 5/9), (1/3, 1), x)
 In:  f19.section(8)
Out:  5*hypergeometric((4/9, 5/9, 10/9), (1, 4/3), x)
 In:  f19.section(10)
Out:  0
```

Dwork Relations. From what precedes, one can write a hypergeometric functions as a $\mathbb{F}_p[x]$-linear combination of p-th powers of other hypergeometric functions. We call these relations *Dwork relations* and they are implemented in the method `dwork_relation()`. Here the output is a dictionary, where hypergeometric functions are assigned polynomial coefficients: the sum of the p-th power of the keys, weighted by the assigned values gives the original hypergeometric function.

```
 In:  f19.dwork_relation()
Out:  {hypergeometric((1/9, 4/9, 5/9), (1/3, 1), x):
       8*x^2 + 14*x + 1,
       hypergeometric((4/9, 5/9, 10/9), (1, 4/3), x):
       5*x^8 + 15*x^7}
```

Annihilating Polynomials. An annihilating polynomial of a reduction of a hypergeometric function can be computed with the method `annihilating_ore_polynomial()`, which outputs a polynomial in the Frobenius: to view it as an actual polynomial, one should replace `Frob`^{n} by X^{p^n}.

```
 In:  f19.annihilating_ore_polynomial()
Out:  (18*x^76 + 13*x^57 + 6*x^38 + 17*x^19 + 12)*Frob^2 +
       (12*x^38 + 11*x^32 + ... + 18*x^12 + 7)*Frob +
       x^30 + 16*x^29 + ... + 6*x^13 + x^12
```

An algorithm performing this computation, based on iteratively computing Dwork relations that lead to a system of equations between finitely many hypergeometric functions, was described in [5, §3.3].

This algorithm is implemented essentially. We warn the user that the current implementation relies on a simplified version of the algorithm, for which it is not proven that the computed system only involves finitely many hypergeometric series, so it might happen that computations do not terminate. However, we believe that also the current implementation can be shown to always terminate.

The polynomials obtained this way are never irreducible. By their nature as linearized polynomials, they are always divisible by $h(x)$. However, in general, they will factor even further.

Congruences Modulo Primes. It is possible that two hypergeometric functions with different set of parameters leads to series which are congruent modulo p, as showcased in the code example below.

```
In:   T.<y> = GF(13)[]
In:   h1 = hypergeometric([1/12, 1/4], [1/2], y)
In:   h2 = hypergeometric([1/12, 1/6], [1/3], y)
In:   h1.power_series(1000) - h2.power_series(1000)
Out:  O(y^1000)
```

The method `is_equal_as_series()` checks when this happens.

```
In:   h1.is_equal_as_series(h2)
Out:  True
```

We sketch an algorithm to check when this holds. Our method relies on the following basic observation: two series are congruent modulo p if and only if all their r-th sections for $0 \leq r < p$ are congruent modulo p. Besides, in our case of interest, we can compute the sections of an hypergeometric series, which we have seen to be constant multiples of hypergeometric functions themselves. We can then proceed recursively. Reusing the arguments we used for `annihilating_polynomial()`, we conclude that we will only encounter finitely many hypergeometric series while proceeding.

To implement this strategy we rely on two auxiliary sets for recording the progress of the algorithm, namely

- the set Q (for "queued") of pairs $(f_1(x), f_2(x))$ whose congruence modulo p still needs to be checked, and
- the set C (for "checked") of pairs $(f_1(x), f_2(x))$ whose congruence modulo p has already been checked.

At first glance, we only transfer the problem to deciding whether a finite number of pairs of hypergeometric functions have the same reduction. However, by checking that all sections of a pair of such functions have the same constant term, we check that the pair of functions agree up to order x^{p-1}. Thus, by only checking equality of *constant terms,* we can decide equality: if for any pair encountered the constant terms do not agree, the reductions of the two hypergeometric functions are distinct, otherwise we iterative conclude equality up to an arbitrary order of precision, *i.e.*, the reductions coincide.

p-curvatures. For hypergeometric functions with $m = n - 1$, we implement the p-curvature of the associated hypergeometric differential operator $\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\beta}; x)$, i.e., a matrix representation of linear map ∂^p acting on $\mathbb{F}_p[x]\langle\partial\rangle/\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\beta}; x)\mathbb{F}_p[x]\langle\partial\rangle$.

It can be accessed by the method `p_curvature()`. Its corank determines the $\mathbb{F}_p(x^p)$ dimension of solutions of the differential operator $\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\beta}; x)$ in $\mathbb{F}_p(x)$.

Algorithm 1: `are_congruent`

```
input : $h_1(x), h_2(x), p$
output: whether $h_1(x) \equiv h_2(x) \pmod p$
$Q \leftarrow \{(h_1(x), h_2(x))\}$;
$C \leftarrow \emptyset$;
while $Q \neq \emptyset$ do
    pop a pair $(f_1(x), f_2(x))$ from $Q$;
    if $(f_1(x), f_2(x)) \in C$ or $(f_2(x), f_1(x)) \in C$ then
        continue;
    for $r \leftarrow 0$ to $p-1$ do
        for $i = 1, 2$, write $\Lambda_r(f_i(x))$ mod $p$ as $a_i x^{e_i} g_i(x)$
          with $a_i \in \mathbb{F}_p$, $e_i \in \mathbb{N}$ and $g_i(x)$ hypergeometric;
        if $a_1 x^{e_1} \neq a_2 x^{e_2}$ in $\mathbb{F}_p[x]$ then
            return false;
        if $a_1 x^{e_1} \neq 0$ in $\mathbb{F}_p[x]$ then
            append $(g_1(x), g_2(x))$ to $Q$;
    append $(f_1(x), f_2(x))$ to $C$;
return true;
```

The corank of the p-curvature for a given hypergeometric varies in p uniformly [5, Prop. 3.1.20] for large enough primes p. The corresponding congruence classes and exceptions are implemented via the method `p_curvature_coranks()` for hypergeometric functions defined over $\mathbb{Q}$.

```
In:   f5 = f % 5
In:   f5.p_curvature()
Out:  [              0 2/(x^5 + 4*x^4) 1/(x^4 + 4*x^3)]
      [              0               0               0]
      [              0               0               0]
In:   f.p_curvature_coranks()
Out:  {1: Empty set of prime numbers,
       2: Set of all prime numbers with 3 excluded:
          2, 5, 7, 11, ...,
       3: Empty set of prime numbers}
```

5 Hypergeometric functions over $\mathbb{Q}_p$

Last, we deal with hypergeometric functions with rational parameters defined over the field of p-adic numbers $\mathbb{Q}_p$. For $x \in \mathbb{Q}_p$, we let $\mathrm{val}_p(x)$ denote its p-adic valuation and we let $\|x\|_p = p^{-\mathrm{val}_p(x)}$ be its p-adic norm.

Radius of Convergence. Similar to the complex case, the p-adic radius of convergence of a series $s(x) = \sum a_k x^k \in \mathbb{Q}_p[[x]]$ is defined as

$$\liminf_{k\to\infty} \|a_k\|_p^{-1/k}.$$

When $\|a\|_p$ (with $a \in \mathbb{Q}_p$) is less than this critical value, the series $s(a)$ converges in $\mathbb{Q}_p$. The method `log_radius_of_convergence()` computes the logarithm in base p of the p-adic radius of convergence of a hypergeometric series.

```
In:   hp5 = h.change_ring(Qp(5))
In:   hp5.log_radius_of_convergence()
Out:  -7/2
```

The algorithm for computing the logarithmic p-adic radius of convergence of ${}_nF_m\,(\boldsymbol{\alpha}, \boldsymbol{\beta}; x)$ closely follows the discussion of [4, §2]. We first partition $\boldsymbol{\alpha} = \boldsymbol{\alpha}' \sqcup \boldsymbol{\alpha}''$, $\boldsymbol{\beta} = \boldsymbol{\beta}' \sqcup \boldsymbol{\beta}''$, where $\boldsymbol{\alpha}', \boldsymbol{\beta}'$ contain precisely the p-adic integers among the parameters. Then, assuming that ${}_nF_m\,(\boldsymbol{\alpha}, \boldsymbol{\beta}; x)$ is not a polynomial, its logarithmic radius of convergence is given by the explicit formula

$$\frac{n'-m'-1}{p-1} + \sum_{\alpha\in\boldsymbol{\alpha}''} \mathrm{val}_p(\alpha) - \sum_{\beta\in\boldsymbol{\beta}''} \mathrm{val}_p(\beta),$$

where n' and m' are the cardinalities of $\boldsymbol{\alpha}'$ and $\boldsymbol{\beta}'$ respectively. On the contrary, when ${}_nF_m\,(\boldsymbol{\alpha}, \boldsymbol{\beta}; x)$ is a polynomial, the logarithm radius of convergence is of course infinite.

Valuations. For $v \in \mathbb{Q}$, we call

$$\mathrm{val}_{p,v}(s(x)) \coloneqq \min_{k\geq 0} \mathrm{val}_p(a_k) + vk$$

the *v-drifted p-adic valuation* of the series $s(x)$. For $v = 0$, it clearly coincides with the p-adic Gauss valuation of $s(x)$, and for arbitrary v, it can be interpreted as the p-adic valuation of $s(x)$ on a disk of p-adic radius p^v centered at 0. In particular, it is $-\infty$ when v is greater than the logarithmic p-adic radius of convergence.

The method `valuation()` computes the p-adic valuation of a hypergeometric function, and passing a parameter v, it computes the v-drifted p-adic valuation. Additionally one can also pass the option `position=True`, to also output the minimal index k, for which the valuation is attained for the coefficient in x^k.

```
 In:   fp5 = f.change_ring(Qp(5))
 In:   fp5.valuation()
Out:   0
 In:   hp3 = h.change_ring(Qp(3))
 In:   hp3.valuation(position=True)
Out:   (-4, 2)
 In:   hp5.valuation()
Out:   -Infinity
 In:   hp5.valuation(-7/2)
Out:   0
```

The authors described in [4, §2.2] an algorithm how to compute the v-drifted p-adic valuations of hypergeometric series. It relies on a recursion over the tropical semi-ring and the Floyd-Warshall algorithm to compute the weak transitive closure of a tropical matrix. Keeping track of the minimal index k, for which the valuation is attained for the coefficient h_k is easily possible, as explained in [4, Rem. 2.6].

p-adic Evaluations. When a is a p-adic number with norm less than the radius of convergence, the value $h(a)$ makes sense. Our package allows to compute it using the following obvious syntax.

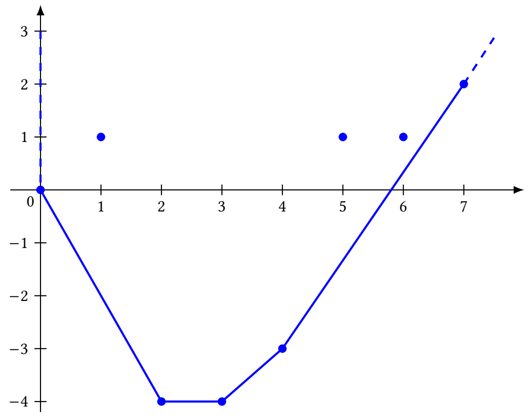
```
 In:   fp5(5)
Out:   1 + 3*5^2 + 5^4 + ... + O(5^20)
 In:   hp3(1/3)
Out:   3^-5 + 2*3^-1 + 1 + 2*3 + ... + O(3^13)
```

The implemented algorithm is outlined in [4, §2.4]: given a p-adic number $a \in \mathbb{Q}_p$, with $\mathrm{val}_p(a) = v$ within the radius of convergence, and a precision N, we first compute a bound K, such than $\mathrm{val}_p(h_k a^k) > N$ for all $k > K$. Thus $h(a) = \sum_{k=0}^{K} h_k a^k + O(p^N)$. The computation of K depends on the choice of a parameter between v and the radius of convergence; the heuristics of our choice is also explained in *loc. cit.*

Newton Polygons. The Newton polygon of a series $\sum_k a_k x^k$ is the convex hull in $\mathbb{R}^2$ of the points (k, v) with $v \geq \mathrm{val}_p(h_k)$.

For hypergeometric series, it can be computed with the method `newton_polygon()`. Passing an argument v, with v chosen smaller than the logarithmic p-adic radius of convergence, can help to handle cases where the Newton polygon has an infinite number of slopes. It shrinks the domain of definition of the hypergeometric series to provide approximations of the Newton polygon.

```
 In:   hp3.newton_polygon()
Out:   ValueError: infinite Newton polygon; try to truncate
        it by giving a log radius less than 2
 In:   NP = hp3.newton_polygon(7/4)
 In:   NP
Out:   Infinite Newton polygon with 5 vertices:
        (0, 0), (2, -4), (3, -4), (4, -3), (7, 2)
        ending by an infinite line of slope 7/4
 In:   NP.plot() +
        point([(i, hp3[i].valuation()) for i in range(8)])
```



We implemented the algorithm of [4, §2.3], which is basically a generalization of the algorithm to compute drifted valuations.

References

[1] F. Beukers and G. Heckman. 1989. Monodromy for the hypergeometric function ${}_nF_{n-1}$. *Inventiones Mathematicae*, 95, 2, 325–354. DOI: 10.1007/BF01393900.

[2] X. Caruso. 2025. Subsets of primes defined by congruence conditions. https://github.com/sagemath/sage/pull/41122.

[3] X. Caruso and F. Fürnsinn. 2025. Algebraic and modular properties of hypergeometric functions. https://github.com/sagemath/sage/pull/41113.

[4] X. Caruso and F. Fürnsinn. 2026. Algorithms for algebraic and arithmetic attributes of hypergeometric functions. eprint: arXiv:2601.16105 (math.NT).

[5] X. Caruso, F. Fürnsinn, and D. Vargas-Montoya. 2025. Galois groups of reductions modulo p of D-finite series. eprint: arXiv:2504.09429 (math.NT).

[6] G. Christol. 1986. Fonctions et éléments algébriques. *Pacific Journal of Mathematics*, 125, 1, 1–37. DOI: 10.2140/pjm.1986.125.1.

[7] G. Christol. 1986. Fonctions hypergéométriques bornées. *Groupe de travail d'analyse ultramétrique*. exp. no 8 14, 1–16. http://www.numdam.org/item/GAU_1986-1987__14__A4_0.pdf.

[8] F. Fürnsinn and S. Yurkevich. 2024. Algebraicity of hypergeometric functions with arbitrary parameters. *Bulletin of the London Mathematical Society*, blms.13103. DOI: 10.1112/blms.13103.

[9] D. Vargas-Montoya. 2024. Algebraicity modulo p of generalized hypergeometric series ${}_nF_{n-1}$. *Journal of Number Theory*, 259, 273–321. DOI: 10.1016/j.jnt.2024.01.004.

[10] D. Vargas-Montoya. 2021. Algébricité modulo p, séries hypergéométriques et structures de frobenius fortes. *Bulletin de la Société mathématique de France*. DOI: 10.24033/bsmf.2834.