On regular and logarithmic solutions of ordinary linear differential systems*

S.A. Abramov[1], M. Bronstein[2], and D.E. Khmelnov[1]

[1] Dorodnicyn Comp. Center of the Russ. Acad. of Sciences, Moscow 119991, Russia,
{sabramov,khmelnov}@ccas.ru
[2] INRIA – Café, BP 93, 06902-Sophia Antipolis Cedex, France

Abstract. We present an approach to construct all the regular solutions of systems of linear ordinary differential equations using the desingularization algorithm of Abramov & Bronstein (2001) as an auxiliary tool. A similar approach to find all the solutions with entries in $C(z)[\log z]$ is presented as well, together with a new hybrid method for constructing the denominator of rational and logarithmic solutions.

1 Introduction

Let C be an algebraically closed field of characteristic 0, z be an indeterminate over C, and

$$L = Q_\rho(z)D^\rho + \cdots + Q_1(z)D + Q_0(z), \tag{1}$$

where $D = d/dz$ and $Q_\rho(z), \ldots, Q_0(z) \in C[z]$. A *regular solution* of $Ly = 0$ (or of L) at a given point $z_0 \in C$, is a solution of the form $(z - z_0)^\lambda F(z)$ with $F(z) \in C((z - z_0))[\log(z - z_0)]$, where $C((z - z_0))$ is the field of (formal) Laurent series over C. If $F(z)$ has valuation 0, then λ is called the *exponent* of the regular solution (otherwise it is an exponent modulo $\mathbb{Z}$). Using the change of variable $\bar{z} = z - z_0$, we can assume without loss of generality that $z_0 = 0$. The problem of constructing all the regular solutions is solved by the Frobenius algorithm (1873, [8, Chap.IV],[9],[14, Chap.V]), which is based on the indicial equation $f(\lambda) = 0$ of L at 0. Not only the roots of $f(\lambda) = 0$, each taken separately, are substantial for the Frobenius algorithm, but also their multiplicities and whether some roots differ by integers. Later, in 1894, L. Heffter proposed another algorithm to solve the same problem ([10, Kap.II,VIII],[14, Chap.V]). For a given root λ of the indicial equation, Heffter's algorithm constructs a basis (possibly empty) for all the regular solutions with exponent λ. Once λ is fixed, that algorithm does not depend on the multiplicity of λ, nor on the existence of another root at an integer distance from λ. It constructs a sequence $E_0, E_1, \ldots$ of linear differential equations, whose right-hand side contains solutions of the preceding equations. If

$$z^\lambda \left(g_0(z) + g_1(z)\frac{\log z}{1!} + g_2(z)\frac{\log^2 z}{2!} + \cdots + g_m(z)\frac{\log^m z}{m!} \right)$$

* Work partially supported by the ECO-NET program of the French Foreign Affairs Ministry, project No. 08119TG, and by RFBR grant No. 04-01-00757.

is a regular solution of (1) then $g_i(z) \in C((z))$ is a solution of E_i for each i. All the regular solutions of (1) with exponent λ have been found when we reach an equation E_{m+1} that has no nonzero Laurent series solution.

To apply the original Frobenius or Heffter algorithm at an arbitrary singularity of a system of linear differential equations would require transforming the system to a scalar differential equation (*e.g.* by the cyclic vector method). That scalar equation usually has huge coefficients, making this approach quite unpractical. If $z = 0$ is a regular singularity of a first-order system of the form

$$\frac{dY}{dz} = A(z)Y(z), \quad A(z) \in \mathrm{Mat}_N(C(z)) \tag{2}$$

then, it is possible in theory to use a variant of the Frobenius algorithm that can be applied directly [8, p. 136, exercise 13], but this approach cannot be applied to irregular singularities or higher-order systems.

A generalization of Heffter's algorithm for constructing the regular solutions of first order systems of the form (2) is described in [5, §5] (an extended version is in [6, §3]). That algorithm is direct, *i.e.* it does not use any uncoupling procedure. A necessary step is however to find all the Laurent series solutions of a given system. For this task and for producing the indicial equation $f(\lambda) = 0$, the algorithm of [6, 5] transforms the system into its *super-irreducible* form (see [11]).

We describe in this paper another adaptation of Heffter's algorithm to linear differential system, which uses the desingularization algorithm of [2, 3] instead of transforming a system into its super-irreducible form. This allows us to handle higher order systems, *i.e.* operators such as (1) where the Q_i are matrices of polynomials, directly, *i.e.* without converting them to larger first-order systems. We study the efficiency of the approach both from the theoretical and practical viewpoints, and it shows that solving them directly is more efficient.

In a similar way, we solve the related problem of finding all the solutions with entries in $C(z)[\log z]$, which we call *logarithmic* solutions. This problem is decomposed into first finding a universal denominator for the solutions, and then finding solutions with entries in $C[z][\log z]$. The latter problem is solved by a slightly modified version of our algorithm for the regular solutions. For the denominator, in addtion, we propose a new hybrid method that combines the algorithm of [2] with a reduction algorithm specific to regular singularities [7]. The hybrid method is applicable to the case of first order systems, and in this case it speeds up the computation quite often (see Sect. 5).

2 Desingularization of linear recurrence systems

Linear recurrences with variable coefficients are of interest for many applications (*e.g.* combinatorics and numeric computation). Consider a recurrence of the form

$$P_l(n)x_{n+l} + P_{l-1}(n)x_{n+l-1} + \cdots + P_t(n)x_{n+t} = r_n \tag{3}$$

where $l \geq t$ are arbitrary integers, $x = (x^1, \ldots, x^N)^T$ is a column vector of unknown sequences (such that $x_i = (x_i^1, \ldots, x_i^N)^T$), $P_t(n), \ldots, P_l(n) \in \mathrm{Mat}_N(C[n])$,

$P_t(n) \neq 0 \neq P_l(n)$ and $r_n \in C[n]^N$. The matrices $P_l(n)$ and $P_t(n)$ are called respectively the *leading* and *trailing* matrices of the recurrence. When $P_t(n)$ and $P_l(n)$ are nonsingular, the roots of their determinants are important for determining the structure of the solution space, as they give bounds on the solutions whose support is bounded above or below. It may happen however that $P_t(n)$ or $P_l(n)$ is singular (or both). In that case, they do not yield bounds on the solutions, but it is also difficult, from a computational standpoint, to use the recurrence (3) to compute the sequence of vectors that it generates. A natural solution in that case is to transform the recurrence system into an equivalent one with either the leading or trailing matrix nonsingular. That transformation may be a "quasi–equivalence", in the sense that the eventual changes in the solution space can be easily described. Such a transformation (the EG-algorithm) was developed in [1] and later improved in [2]. In addition to the transformed system, it also yields a finite set of linear constraints such that the solutions of the original system are exactly those of the transformed system that also satisfy the new constraints (each of the constraints is a linear relation that contains a finite set of variables x_i^j).

3 Regular solutions

We consider in this section the higher order system $LY = 0$ where L is of the form (1) with $Q_0, \ldots, Q_\rho \in \mathrm{Mat}_N(C[z])$ and Q_ρ nonsingular.

3.1 Description of the algorithm

Using the standard basis $(z^m)_{m \geq 0}$ of $C[z]$, we construct (see [2, §2]) its associated recurrence system $Rc = 0$, where $R = P_l(n)E^l + \cdots + P_t(n)E^t$, E is the shift operator and $P_j(n) \in \mathrm{Mat}_N(C[n])$ for $t \leq j \leq l$. If $\det P_l(n)$ is identically 0, then it is possible (see Section 2) to transform the recurrence system into an equivalent one (together with a finite set of linear constraints) with $\det P_l(n) \neq 0$, so assume from now on that $\varphi(n) = \det P_l(n) \neq 0$. Let $\psi(n) = \varphi(n-l)$ and n_0, n_1 be respectively the minimal and maximal integer roots of $\psi(n)$ (if there is no integer root, then $LY = 0$ has no Laurent series solution). Any Laurent series solution of $LY = 0$ has no term $c_k z^k$ with $c_k \in C^N$ and $k < n_0$. Using the recurrence $Rc = 0$ and the additional constraints, we can, by a linear algebra procedure, compute a basis of the linear space of initial segments $c_{n_0} z^{n_0} + c_{n_0+1} z^{n_0+1} + \cdots + c_M z^M$, where M is a fixed integer, chosen greater that n_1 and all the indices appearing in the linear constraints. Observe that if our differential system is inhomogeneous with a Laurent series right-hand side (whose coefficients are given by a linear recurrence system), then we can similarly construct a basis of the affine space of its Laurent series solutions. If $\psi(n)$ has a non-integer root λ, then the preliminary change of variable $Y = z^\lambda \bar{Y}$ produces a new system for $\bar{Y}$, hence a new recurrence with a new $\bar{\psi}(n) = \psi(n - \lambda)$. Therefore, we can always work with the integer roots of ψ. For any integer $m \geq 0$, the result of applying L to $g(z)\log^m(z)/m!$

is clearly of the form

$$L_{m,m}(g)\frac{\log^m z}{m!} + \cdots + L_{m,1}(g)\frac{\log z}{1!} + L_{m,0}(g), \tag{4}$$

where the coefficients of the differential operators $L_{i,j}$ belong to $\mathrm{Mat}_N(C(z))$. Proofs of the following proposition can be found in [10] and [12, Sect. 3.2.1].

Proposition 1 *The coefficients of all the $L_{i,j}$ in (4) belong to $\mathrm{Mat}_N(C[z,z^{-1}])$. In addition, $L_{0,0} = L$ and $L_{i+j,j} = L_{i,0}$ for any $i,j \geq 0$.*

Let $L_i = L_{i,0} (= L_{i+j,j}$ for any $j \geq 0)$. Using (4) and Proposition 1 we obtain

$$L\left(\sum_{m=0}^{k} g_{k-m}(z)\frac{\log^m z}{m!}\right) = \sum_{m=0}^{k}\left(\sum_{j=0}^{k-m} L_j(g_{k-m-j})\right)\frac{\log^m z}{m!}\,.$$

Therefore,

$$Y = \sum_{m=0}^{k} g_{k-m}(z)\frac{\log^m z}{m!} \tag{5}$$

is a solution of $LY = 0$ if and only if $(g_0(z), \ldots, g_k(z))$ is a Laurent series solution of the inhomogeneous linear system

$$L_0(g_i) = -\sum_{j=1}^{i} L_j(g_{i-j}) \quad \text{for } 0 \leq i \leq k\,. \tag{6}$$

When we find $g_0(z)$ using the first equation $L_0(g_0) = 0$ of (6), that solution contains arbitrary constants. When we use $g_0(z)$ in the right-hand side of the next equation $L_0(g_1) = -L_1(g_0)$ of (6) those arbitrary constants appear linearly in the right-hand side. Using the same technique as when solving such scalar parametric inhomogeneous equations (see for example [4]), we find together with $g_1(z)$ linear constraints on the arbitrary constants appearing in g_0 and g_1. Repeating this process, we find at each step that $g_0, \ldots, g_i$ depend on unknown constants together with a linear system for those constants. In order for this process to terminate, we need to ensure that we always reach an integer k such that (6) has no Laurent series solution with $g_0 \neq 0$. Heffter proved this in the scalar case, and his proof carries over to systems.

Proposition 2 *The set $K = \{k \geq 0$ such that (6) has a solution with $g_0 \neq 0\}$ is finite. If K is empty, then $LY = 0$ has no nonzero solution in $C((z))[\log z]$. Otherwise, $K = \{0, \ldots, \mu\}$ for some $\mu \geq 0$ and any solution in $C((z))[\log z]$ of $LY = 0$ has the form (5) where $(g_0, \ldots, g_\mu)$ is a solution of (6) with $k = \mu$. In addition, any solution of (6) with entries in $C((z))$ generates a solution of $LY = 0$.*

Proof. Let G_k be the linear space of all the (regular) solutions of the form (5) of $LY = 0$, and $Y \in G_k$. Writing $Y = \sum_{m=0}^{k+1} h_{k+1-m}(z)\log^m(z)/m!$ where $h_0 = 0$

and $h_{i+1} = g_i$ for $0 \le i \le k$, we see that $G_0 \subseteq G_1 \subseteq \cdots \subseteq G_k \subseteq G_{k+1} \subseteq \cdots$. Let $k > 0$ be in K and $(g_0, \ldots, g_k)$ be a solution of (6) with $g_0 \ne 0$. Then, $(g_0, \ldots, g_{k-1})$ is a solution of (6) with $k-1$ and so on, which implies that $\{0, \ldots, k\} \subset K$ and that $\dim_C G_k \ge k$. This produces k linearly independent solutions of the form (5) of $LY = 0$. On the other hand, since $LY = 0$ is equivalent to a scalar equation of order at most ρN, $\dim_C G_k \le \rho N$ for all k, so K is either empty or of the form $\{0, \ldots, \mu\}$ for some $\mu \le \rho N$. Therefore, if we compute $G_0, G_1, \ldots$ using (6), we eventually find an integer $k \le \rho N$ for which the system (6) has a solution only for $g_0 = 0$. At this point, G_{k-1} contains all the solutions of $LY = 0$ with entries in $C((z))[\log z]$.

Summarizing, our scheme for constructing regular solutions of $LY = 0$ is:

1. Construct its associated matrix recurrence in the form (3) and transform it into an equivalent one with nonsingular leading matrix (see Section 2). Let

$$P'_l(n) z_{n+l} + P'_{l-1}(n) z_{n+l-1} + \cdots + P'_t(n) z_{n+t} = r'_n \tag{7}$$

   be the resulting recurrence (it may include in addition a set of linear constraints). Compute all roots of $\varphi(n) = \det(P'_l(n))$, divide them into groups having integer differences, and construct the set Λ consisting of one representative for each groups.
2. For each $\lambda \in \Lambda$, compute regular solution whose exponent is λ:
   (a) Compute a system S_λ by substituting $Y = z^\lambda Y_\lambda$ and by following multiplication of the system by $z^{\rho-\lambda}$. This induces a transformation of (7). We get a recurrence R_λ: $P''_l(n) z_{n+l} + P''_{l-1}(n) z_{n+l-1} + \cdots + P''_t(n) z_{n+t} = r'_n$, where $P''_i(n) = P'_i(n + \lambda)$, $i = l, l-1, \ldots, t$, and since $P'_l(n)$ is nonsingular, so is $P''_l(n)$ and no additional desingularisation is required. The transformed recurrence R_λ may include a set of linear constraints which are transformed correspondingly.
   (b) Determine the number M_λ of required initial terms of Laurent series, which is greater than all the integer roots of the determinant of the leading matrix of R_λ as well as all the indices appearing in the additional constraints.
   (c) Successively solve systems (6) for the required number of terms of its Laurent series solutions, using the recurrence R_λ while it is possible. This yields regular solutions y_λ of S_λ in the form (5).
3. Combine all the solutions into the general regular solution $y = \sum_{\lambda \in \Lambda} z^\lambda y_\lambda$.

Note that we construct regular solutions by representing all involved Laurent series by truncated expansions until an appropriate terms, such that the number of linearly independent solutions is determined correctly and computation of the subsequent terms can be performed one by one by means of recurrenses $\{R_\lambda\}$.

3.2 Computing and implementation remarks

The associated recurrence system The main part of the algorithm is solving the individual systems from the sequence (6). Note however that all those systems have in the left-hand side the same operator $L_0 = L$. Therefore, the associated linear recurrence systems also have the same left-hand sides, but their right-hand sides are different. In order to desingularize the recurrence system only once, we apply during the first desingularization (step 1 above) all the transformations to a generic right-hand side. As a result, we have the transformed recurrence with a non-singular leading matrix, a finite set of linear constraints and the transformed right-hand side in a generic form. Each component of this generic transformed right-hand side is a linear combination of possibly shifted components of the original right-hand side before desingularization. In this way we can use the same transformed recurrence for solving any system from the sequence (6) specifying the concrete right-hand side by substituting the corresponding values into the generic right-hand side (such substitutions commute with any elementary transformation that our desingularization algorithm uses).

Computing the right-hand sides This is not so simple since the right-hand side for the i-th system in (6) is $h_i = -\sum_{j=1}^{i} L_j(g_{i-j})$, where $g_0, \ldots, g_{i-1}$ are Laurent series solutions of the preceding systems. Since we represent (truncated) Laurent series solution by segment of initial terms, we need to determine the required numbers of initial terms of $g_0, \ldots, g_{i-1}$. That number is determined in an algorithmic way and depends on the number M_λ of initial terms of the transformed right-hand side, which is determined in step 2b of the algorithm for all the systems in the sequence (6), and ensures that the next terms of the series are computed from the preceding ones by a simple use of the recurrence. So we compute the transformed right-hand side in the following way:

1. Taking into account M_λ and the components of the transformed generic right-hand side, compute the numbers of required initial terms of the components of the right-hand side before transformation, to ensure that the number of initial terms in the transformed right-hand side is equal to M_λ.
2. Taking into account the form of the operators $L_1, \ldots, L_i$, compute the numbers of initial terms of $g_0, \ldots, g_{i-1}$ required to ensure the needed numbers of initial terms of the components of right-hand side before transformation.
3. Compute the corresponding initial segments of $g_0, \ldots, g_{i-1}$.
4. Compute the initial segment of the right-hand side before transformation substituting the initial segments of $g_0, \ldots, g_{i-1}$ into h_i.
5. Compute the initial segment of the transformed right-hand side substituting the initial segment of the right-hand side before transformation into the transformed generic right-hand side.

Extending solution components Computing the transformed right-hand side depends on computing the initial segments of $g_0, \ldots, g_{i-1}$ (step 3 in Section 3.2). Since the required number of initial terms of the solution component g_k may be greater than the number of the initial terms computed in the preceding steps of the algorithm, we need to extend that component, which requires computing additional terms using the associated recurrence. This means in turn that we need to extend the corresponding transformed right-hand side, computing it using the approach from Section 3.2 while replacing M_λ by the new number. Note that this may again require extending other solution components, so this procedure is recursive.

Computing initial segments When we compute the transformed right-hand side of the recurrence, solving a system from the sequence (6) for the required number of initial terms can be done step by step using the recurrence. In each step one of the following options occurs:
- the next term is computed as a function of the previous terms;
- a linear constraint on the previous terms appears, which can be either solved or inconsistent, in which case there is no Laurent series solution;
- the next term is a new arbitrary constant (which may be specified later in the computation when solving constraints).

After all those steps, either all the initial terms have been computed (some of them being arbitrary constants) or we have proven that there is no Laurent series solution. Note that:

1. Since each of the $g_0, \ldots, g_{i-1}$ may have arbitrary constants, the right-hand side for the i-th system in the sequence (6) may have the same arbitrary constants. This leads to the fact that, during the computation of the Laurent series solution of the i-th system, some of the arbitrary constants may be specified when resolving newly appearing constraints. This could turn the current initial segment of g_0 to zero. Since the number of terms in the segment is such that all the remaining terms are computed by the associated recurrence whose leading matrix is non-singular, this implies that g_0 is identically zero. As noted earlier, when this happens, then all the solution components have been computed and i-th system has no Laurent series solutions with $g_0 \neq 0$.
2. As noted earlier, since $G_k \subset G_{k+1}$, we can use only the last solution found of the form (5) with $g_0 \neq 0$ as the regular solution whose exponent is λ, since it also contains all the previously found solutions of that form.

4 Logarithmic solutions

Finding regular solutions is a local problem. We consider in this section the analogous global problem of finding solutions whose entries are in $C(z)[\log z]$ of the higher order system $LY = 0$ where L is of the form (1) with $Q_0, \ldots, Q_\rho \in \mathrm{Mat}_N(C[z])$ and Q_ρ nonsingular. In the case of first-order systems $Y' = AY$, an algorithm, based on super-irreducible forms, for computing such solutions was presented in [5]. As for regular solutions, we present here an alternative that does not require conversion to first order systems, and that uses desingularization (see Section 2) instead of super-irreducible forms. Recall that finding rational solutions of $LY = 0$ proceeds in two distinct steps: we construct first a universal denominator, *i.e.* $d \in C[z]$ such that $dY \in C[z]^N$ for any solution $Y \in C(z)^N$

of $LY = 0$. We then search for the polynomial solutions of the system obtained by the change of variable $\bar{Y} = dY$ in L. As remarked in [5, §5.1], a universal denominator for rational solutions is also valid for logarithmic solutions. As described in [2, §8.1], the irreducible factors of a universal denominator must all divide $\det(Q_\rho)$ and we can compute such a universal denominator using the algorithm described there. This reduces our problem to finding solutions whose entries are in $C[z][\log z]$. Before solving that problem in Section 4.2, we first present a heuristic for accelerating the computation of universal denominators in the case of first-order systems.

4.1 A hybrid heuristic for first-order systems

We restrict in this section our systems to be of the form (2). If all the finite singularities of that system are known to be regular, then the reduction method of [7] transforms all of them *simultaneously* into simple poles, that is, it returns a change of variable that transforms the system into an equivalent one where the denominator of A is squarefree. In that case, the exponents at all the singularities can be easily computed as eigenvalues of the associated residue matrices, yielding a fast way to compute a universal denominator. That algorithm repeats the following *single reduction step*: as long as the denominator of A is not squarefree, choose a row of A whose common denominator $d \in C[z]$ is not squarefree. Using only extended gcd computations, compute an invertible $T \in \mathrm{Mat}_N(C(z))$ whose rows form a $C[z]$-basis of the free $C[z]$-module $C[z]^{1\times N} + C[z]\omega$ where $C[z]^{1\times N}$ is the module of row-vectors and ω is the selected row of A multiplied by the squarefree part of d. Apply then the change of variable $\bar{Y} = TY$ to $dY/dz = AY$ and repeat this process, which is shown in [7] to yield a matrix with a squarefree denominator after finitely many steps.

Not much is known about the behavior of that algorithm in the presence of irregular singularities (except that it cannot terminate!) but we use it here as a heuristic to separate the (proven) regular and (putative) irregular singularities of the system. Since the proven regular singularities are transformed into simple poles in this process, we use the eigenvalue approach to compute their exponents, limiting the use of desingularization to the remaining ones. Note that wrongly classifying a singularity as irregular does not affect the correctness of the hybrid method, since going through the recurrence approach at such singularities yield a correct bound. This yields the following hybrid heuristic for the universal denominator, given a first-order system in the form (2):

1. Initialize the set of proven regular singularities to be $R := \emptyset$ and the universal denominator to be $U := 1$.
2. Compute an irreducible factorisation of the denominator of A, the set F of its irreducible factors and the subset F_1 of factors having multiplicity 1.
3. Initialize the set $S := F$ of "unclassified" singularities.
4. Repeat the following steps:
   (a) For each factor in $f \in F_1$ (proven regular singularities) compute its exponent $e \geq 0$ in the universal denominator by the classical approach for simple poles. Update $U := Uf^e$.

(b) Update $R := R \cup F_1$ followed by $S := S \backslash R$.

(c) Repeat the single reduction step of [7] described above, limiting it to the factors in S (see below), and get the transformed system with matrix A_r until either (i) a new squarefree factor of the denominator of A_r is found, or (ii) we have determined that all the elements of S correspond to putative irregular singularities, using the following heuristic: for each $f \in S$ compute
   - mf — minimal positive multiplicity of f in the denominators of the rows of the original A
   - mf' — the same but for the rows of the transformed A_r
   - sf — sum of the multiplicities of f in the denominators of the rows of the original A
   - sf' — the same but for the rows of the transformed A_r

   We declare f to be a putative irregular singularity if one of the following conditions hold:
   - $mf' > mf$;
   - $mf' = mf$ and $sf' > sf$;
   - $mf' = mf$, $sf' = sf$ and this situation is repeated for the third successive single reduction step (note that we count the number of such successive stabilities of mf and sf and pass that count from one reduction step to the next one)

   If f has been declared to be a putative irregular singularity, then we remove it from S for the next single reduction step.

(d) Let $F_1 \subset F \backslash R$ be the set of factors of the denominator of A_r with multiplicity 1. If $F_1 \neq \emptyset$, then go back to step 4a otherwise leave the loop.

5. For each $f \in F \backslash R$ (putative irregular singularities) compute its exponent $e \geq 0$ in the universal denominator by the desingularization approach of [2] (see Sect. 2). Update $U := U f^e$.

Note that the above algorithm is computing the exponent of factors reduced to simple poles as soon as they appear during the reduction, so we can exclude them from the following reductions. This is more efficient than computing them at the end since each single reduction step increases the degrees and coefficients of the entries of A_r. We can use the exponents computed using the transformed matrix A_r at each step as the correct exponents in a universal denominator for the original matrix A because the change of variables at each single step are given by inverses of polynomial matrices [7, Theorem 2].

To limit the single reduction step to the factors in $S \subset F$ at step 4c, we adapt the single reduction step as follows: the denominator d of each row of A can be factored as $d = \prod_{p \in S} p^{e_p} \prod_{q \notin S} q^{e_q}$. Choose a row of A for which $e_p > 1$ for some $p \in S$, let ω be that row of A multiplied by $\prod_{p \in S} p^{\min(e_p,1)} \prod_{q \notin S} q^{e_q}$ and compute a basis of $C[z]^{1 \times N} + C[z]\omega$ as explained above.

A final remark about the hybrid method: there is some arbitrariness in selecting the row used in the single reduction step. In our implementation, we select the row with the maximal sum of the multiplicities of the factors from S in its denominator.

4.2 Solutions with entries in C[z][log z]

Since our algorithm for finding regular solutions at $z = 0$ returns truncated Laurent series, it can easily be adapted to return only the solutions with entries in $C[z][\log(z)]$. Only the following changes are needed:

- Instead of dynamically bounding the number of terms of the series to compute, we compute an upper bound of the degrees of the polynomial solutions and use that number of terms. Such an upper bound is computed from the roots of the determinant of the *trailing* matrix of the associated recurrence system (see [1, 2] for details).
- As we are not interested in the z^λ factors, we skip the computation of the set Λ of exponents modulo $\mathbb{Z}$ at $z = 0$.

This yields the following algorithm for solutions in $C[z][\log z]$:

1. Construct its associated matrix recurrence in the form (3) and transform it into an equivalent one R with $P_t(n)$ nonsingular (see Section 2).
2. Compute an upper bound M on the degree of the polynomial solutions from the integer roots of $\det P_t(n)$.
3. Successively solve systems (6) for their polynomial solutions, using the recurrence R while it is possible. This means setting all the coefficients of the series with negative indices to 0 and computing truncated series up to the degree bound. As we compute successive right-hand sides, we can refine our degree bound. Since the polynomials are computed completely, there is no need to extend them as we compute additional right-hand sides. As we used a recurrence with nonsingular trailing matrix to bound the degree, it is natural to use it again in this step to compute the coefficients of the polynomials. So we compute them starting with the coefficients of highest degree and work down to the constant coefficients. For regular solutions, we used a nonsingular leading matrix to get the exponents and precisions, so we obtained the coefficients in the reverse order.

5 Complexity and experimental comparisons

Consider the higher order system $LY = 0$ where L is of the form (1) with $Q_0, \ldots, Q_\rho \in \mathrm{Mat}_N(C[z])$ and Q_ρ nonsingular. Let δ be a bound on the degrees of the entries of the Q_i's. There is no known complete complexity analysis for our method, nor for the method of [6]. However, since our method computes only one desingularisation (see Sect. 3.1) and the method of [6] computes only one super-irreducible form, we attempt here to compare those operations.

The basic operation for desingularisation is computing ranks and nullspaces of matrices of polynomials. Using [15], this has a complexity of $\mathcal{O}^\sim(n^\omega d)$ operations in C, where n is the size of the matrix, d a bound on the degree of its entries, ω the exponent of matrix multiplication and the $\mathcal{O}^\sim$ notation means that we ignore logarithmic factors. The leading matrix of our recurrence is of size N and its entries are of degree bounded by ρ (and not δ since the transformation from differential equations to recurrences sends D to n), so the first nullspace has a cost of $\mathcal{O}^{\sim}(N^{\omega}\rho)$.

The basic operation for super-irreducible forms [11] is computing characteristic polynomials of matrices of polynomials. Using [13] this has a a complexity of $\mathcal{O}^{\sim}(n^{\omega+1/3}d)$ operations in C. Since $LY = 0$ must first be converted to a first order system of size $N\rho$, the first characteristic polynomial has a cost of $\mathcal{O}^{\sim}(N^{\omega+1/3}\rho^{\omega+1/3}\delta)$.

In practice, we expect desingularization and super-irreducible forms to converge quickly, *i.e.* to compute relatively few nullspaces or characteristic polynomials. In theory, desingularization can require $N(\delta+\rho)$ nullspaces in the worst case. Since we do not know the growth of the degrees of the entries, this only yields the approximate complexity of $\mathcal{O}^{\sim}(N^{\omega+1}(\delta+\rho)F(\rho))$ where $F(\rho)$ encodes the growth in the degrees. Similarly, super-irreducible forms can require $s(s+1)/2$ characteristic polynomials in the worst case, where s is the polar order of the matrix at $z = 0$, which can be as high as δ. This then yields an approximate complexity of $\mathcal{O}^{\sim}(N^{\omega+1/3}\rho^{\omega+1/3}G(\delta)^3)$ where $G(\delta)$ encodes the growth in the degrees.

We have implemented our regular solution algorithm in Maple on top of the package `LinearFunctionalSystems`, which contains solvers based on [2]. Our function returns the regular solutions with truncated Laurent series of a linear differential system with polynomial coefficients. The number of terms of the series is determined automatically to ensure that the remaining terms of the series can be computed using the associated recurrences (*i.e.* its leading matrix is invertible for all the remaining terms). In order to extend initial segments of the Laurent series, another function is provided, which returns the regular solution with the series extended to a given degree.

For comparison purposes, we used the Maple package `ISOLDE`, which implements the algorithm of [6]. We compared the two programs on several sets of generated systems [3]. Although our program is faster in the majority of examples, the gains for first-order systems are only by a small constant factor. Moreover, since the programs use different approaches, this comparison can only conclude that the two methods are of comparable efficiency for first-order systems and their weak and strong features are displayed on different systems. For exmaple, for one of the sets, most systems were solved faster by our program, but `ISOLDE` had a lower total CPU time for one of the subsets since it solved much faster a few systems in that subset. Those results indicate as well the difficulty of developping a polyalgorithm that would automatically detect the most efficient method to use for each particular input. The main advantage of our method is however its direct applicability to higher-order systems, where the experimental results confirm the better efficiency. For the comparison we generated a set of higher-order systems and as well transformed all the systems in the set to corresponding first order systems. Then we solved the higher-order systems directly by our program and solved the corresponding first-order systems both by our program and by `ISOLDE`. All systems in the set were solved faster by direct approach. To be fair,

[3] All comparisons are available at `www.ccas.ru/sabramov/casc2005.html`

we should remark that our program has been steadily improved by the recent update of some modules, while `ISOLDE` has not been updated for a long time. So we do not exclude the possibility that further improvements in `ISOLDE` could lead to some changes in our comparisons. They nevertheless reflect accurately the current status of those programs, as well as parallel similar conclusions obtained by comparing them on the computation of rational solutions [3].

We have also implemented our logarithmic solution algorithm on top of the package `LinearFunctionalSystems`, together with an option in the corresponding procedure `UniversalDenominator` to allow the use of the hybrid heuristic. Again we generated several sets of systems, and solved them with and without the hybrid heuristic for the universal denominator. The results showed that the hybrid method generally yields a significant speedup, though it might lead to additional work without any gain sometimes.

References

1. Abramov, S.: EG–eliminations. Journal of Difference Equations and Applications **5** (1999) 393–433
2. Abramov, S., Bronstein, M.: On solutions of linear functional systems. In Proceedings of ISSAC'2001, ACM Press (2001) 1–6
3. Abramov, S., Bronstein, M., Khmelnov, D.: Regularization of linear recurrence systems. In Transactions of the A.M. Liapunov Institute. Volume 4. (2003) 158–171
4. Abramov, S., Bronstein, M., Petkovšek, M.: On polynomial solutions of linear operator equations. In Proceedings of ISSAC'95, ACM Press (1995) 290–296
5. Barkatou, M.A.: On rational solutions of systems of linear differential equations. Journal of Symbolic Computation **28** (1999) 547–567
6. Barkatou, M., Pflügel, E.: An algorithm computing the regular formal solutions of a system of linear differential equations. Journal of Symbolic Computation **28** (1999) 569–587
7. Bronstein, M., Trager, B.: A reduction for regular differential systems. In CD-ROM, Proceedings of MEGA'2003. (2003)
8. Coddington, E., Levinson, N.: Theory of ordinary differential equations. McGraw-Hill, New York (1955)
9. Frobenius, G.: Über die Integration der linearen Differentialgleichungen mit veränder Koefficienten. Journal für die reine und angewandte Mathematik **76** (1873) 214–235
10. Heffter, L.: Einleitung in die Theorie der linearen Differentialgleichungen. Teubner, Leipzig (1894)
11. Hilali, A., Wazner, A.: Formes super–irréductibles des systèmes différentiels linéaires. Numerical Mathematics **50** (1987) 429–449
12. van der Hoeven, J.: Fast evaluation of holonomic functions near and in regular singularities. Journal of Symbolic Computation **31** (2001) 717–743
13. Kaltofen, E., Villard, G.: On the complexity of computing determinants. Computational Complexity **13** (2004) 91–130
14. Poole, E.: Introduction to the Theory of Linear Differential Equations. Dover Publications Inc., New York (1960)
15. Storjohann, A., Villard, G.: Computing the rank and a small nullspace basis of a polynomial matrix. In Proceedings of ISSAC'2005, ACM Press (2005)