Algorithms for Regular Solutions of Higher-Order Linear Differential Systems

Moulay A. Barkatou, Thomas Cluzeau, Carole El Bacha
University of Limoges; CNRS; XLIM UMR 6172, DMI
87060 Limoges, France
{moulay.barkatou,thomas.cluzeau,carole.el-bacha}@xlim.fr

ABSTRACT

We study systems of higher-order linear differential equations having a regular singularity at the origin. Using the properties of matrix polynomials, we develop efficient methods for computing their regular formal solutions. Our algorithm handles the system directly without transforming it into a system of first-order but higher size. We give its arithmetic complexity and discuss the results of an implementation in Maple.

Categories and Subject Descriptors

I.1.2 [**Symbolic and Algebraic Manipulation**]: Algorithms

General Terms

Algorithms, Experimentation, Theory

Keywords

Computer algebra, higher-order linear differential systems, regular formal solutions, Frobenius' method, matrix polynomials

1. INTRODUCTION

Let $\mathbb{K}$ be an extension of $\mathbb{Q}$ ($\mathbb{Q} \subseteq \mathbb{K} \subseteq \mathbb{C}$) and $\bar{\mathbb{K}}$ its algebraic closure. We denote by $\mathbb{K}[[x]]$ the ring of formal power series in the variable x and $\mathbb{K}((x))$ its field of fractions. We consider a system of n linear differential equations of order ℓ of Fuchs type

$$\mathcal{L}(y(x)) = \sum_{i=0}^{\ell} A_i(x)\vartheta^i y(x) = 0, \qquad (1)$$

where $\vartheta = x\dfrac{d}{dx}$ is the Euler derivation, for $i = 0, \ldots, \ell$, $A_i(x)$ is a square matrix of size n with entries in $\mathbb{K}[[x]]$ and $y(x)$ is an unknown n-dimensional vector. We assume that $A_\ell(0)$ is invertible which means that System (1) has a regular singularity at the origin. We call *regular formal solution* of (1) a formal solution of the form

$$y(x) = x^{\lambda}\, z(x),$$

where $\lambda \in \bar{\mathbb{K}}$ and $z \in \bar{\mathbb{K}}[[x]]^n[\log(x)]$. In this article, we address the problem of constructing a basis of the regular formal solution space of System (1).

Such systems appear in several applications in physics and engineering sciences: for example, in automatic, a classical approach proceeds by linearizing a given non-linear differential system along a particular trajectory which naturally results in a system of the form (1). They have been studied in applied mathematics (see for example [21] and references therein or [13, 18]).

The standard approach to solve such a system consists in converting it into a first-order system of size $n\,\ell$

$$\vartheta Y = \mathcal{C}Y,$$

where $Y = (y^T, \vartheta y^T, \ldots, \vartheta^{\ell-1}y^T)^T$, and $\mathcal{C}$ is the block companion matrix given by

$$\mathcal{C} = \begin{pmatrix} 0 & I & 0 & \ldots & 0 \\ 0 & 0 & I & \ldots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \ldots & \ldots & I \\ \tilde{A}_0 & \tilde{A}_1 & \ldots & \ldots & \tilde{A}_{\ell-1} \end{pmatrix}, \qquad (2)$$

where I is the identity matrix and for $i = 0, \ldots, \ell-1$, $\tilde{A}_i = -A_\ell^{-1} A_i$. The resulting system can be solved by one of the existing algorithms for first-order systems (see [4, Ch. 2], [8, Ch. 4] or [5]): for example, one can use the algorithm in [5] which solves the problem even in case of an irregular singularity. An implementation of that algorithm can be found in the package Isolde (see [6]). However, this approach has the drawback of increasing the size of the system. Note also that algorithms for first-order systems do not take advantage of the structure of the companion matrix $\mathcal{C}$ and furthermore, the transformations applied break this structure. For those reasons, the idea at the beginning of this work was to look for methods that handle directly System (1).

To our knowledge, there exist only few works on that direction. In [3], Abramov et al prove that the Heffter method [12, Ch. 8] can be generalized for systems of arbitrary order: the problem of finding solutions associated to a given exponent $\lambda \in \bar{\mathbb{K}}$ is reduced to computing Laurent series solutions of auxiliary non-homogeneous higher-order linear differential systems. To find these Laurent series, they compute an associated matrix recurrence equation and use EG-elimination to compute its desingularization (see [1]). The Maple package LinearFunctionalSystems contains an implementation of this method.

In [16], Jódar and Legua propose a generalization of the Frobenius method (see [10], [14, Ch. 16] or [8, Ch. 4, Section 8]) to systems of the form (1) with $\ell = 2$. But this latter work treats only the case where System (1) has only logarithm-free regular solutions, *i.e.*, solutions of form $y = x^{\lambda} z$ with $\lambda \in \bar{\mathbb{K}}$ and $z \in \bar{\mathbb{K}}[[x]]^n$.

Finally, in [17], the authors compute solutions of System (1) of the form $(\sum_{i=0}^{\infty} D_i\, x^i)\, x^Z$ where $D_i \in \bar{\mathbb{K}}^{n\times m}$ and $Z \in \bar{\mathbb{K}}^{m\times m}$ but they only consider the case where the eigenvalues of the matrix $\mathcal{C}(0)$ do not differ by integers.

The first contribution of the present paper is the generalization of the approach proposed by Poole ([20, Ch. 5, § 16]) in the scalar case ($n = 1$) to Systems (1) of arbitrary order ℓ and size n. The case $\ell = 1$ is also of interest since we did not find, in the literature, any work adapting Poole's idea to first-order systems. This method consists in arranging the solution $y(x)$ as a Frobenius series in x which coefficients are polynomials in $t = \log(x)$ of degree less than $n\,\ell$:

$$y(x) = x^{\lambda_0}(U_0(t) + U_1(t)\, x + \cdots + U_i(t)\, x^i + \cdots),$$

with $\lambda_0 \in \bar{\mathbb{K}}$, $U_i(t) \in \bar{\mathbb{K}}[t]^n$ and $U_0 \neq 0$. Plugging $y(x)$ into System (1), one finds that λ_0 and $U_0(t)$ must satisfy

$$L_0(\frac{d}{dt} + \lambda_0)U_0(t) = 0,$$

where $L_0(\lambda)$ designates the matrix polynomial

$$L_0(\lambda) = A_\ell(0)\lambda^\ell + \cdots + A_1(0)\lambda + A_0(0),$$

whose determinant plays the role of the *indicial polynomial*. It follows that λ_0 must be chosen as an eigenvalue of $L_0(\lambda)$ and the coefficients of U_0 as a sequence of corresponding generalized eigenvectors (*Jordan chain*). For $i \geq 1$, we get

$$L_0(\frac{d}{dt} + \lambda_0 + i)\, U_i(t) = P_i(t),$$

where $P_i(t)$ depends only on $U_0(t), \ldots, U_{i-1}(t)$. Thus the problem reduces to computing polynomial solutions of non-homogeneous linear differential systems with constant matrix coefficients. This latter problem is studied in Subsection 2.3 where we prove that such systems always admit at least a polynomial solution and we give a bound on its degree. We then give a method to construct the coefficients U_i based on well-known properties (*e.g.*, Jordan chains, partial multiplicities, ...) of matrix polynomials ([11, Ch. 1]) and we give a complexity estimate of the whole algorithm.

As a second contribution, we also show that, using properties of matrix polynomials, the Frobenius method can be extended to compute the regular solutions of System (1): the determinant of the matrix polynomial $L_0(\lambda)$ playing the role of the indicial polynomial.

Another contribution is that we provide an implementation of these algorithms in Maple. We compare it to the implementation of the algorithm in [3] available in the package LinearFunctionalSystems of Maple and to the standard approach which consists in converting (1) into a first-order system and using Isolde. It turns out that our implementation[1] of Algorithm 2 in Subsection 3.1 gives good results.

[1]available at http://www.ensil.unilim.fr/~cluzeau

The rest of the paper is organized as follows: Section 2 is concerned with the constant coefficients case. We first recall some properties of matrix polynomials and how to use them to compute a basis of the regular formal solution space. We then prove that a non-homogeneous system with a polynomial (in $t = \log(x)$) right-hand side always admits at least a polynomial solution in t. Section 3 contains our main contributions to the problem: we show how to generalize the Poole and Frobenius algorithms to the matrix case. Section 4 concentrates on the extension of the previous algorithms to the case where the matrix $A_\ell(0)$ in System (1) is no longer assumed to be invertible but $A_\ell(x)$ invertible in $\mathbb{K}((x))^{n\times n}$: notice that in this case, System (1) may have an irregular singularity at the origin. Finally, in Section 5, we discuss the results of our implementation of the algorithms developed in the paper and we propose some experimental comparisons.

2. THE CONSTANT COEFFICIENTS CASE

In this section, we study the simplest case where the coefficients of System (1) are constant matrices. Such a system can be written as

$$A_\ell \vartheta^\ell y(x) + \cdots + A_1 \vartheta y(x) + A_0 y(x) = 0, \qquad (3)$$

where for $i = 0, \ldots, \ell$, A_i belongs to $\mathbb{K}^{n\times n}$ and A_ℓ is invertible. In the sequel, Equation (3) will be called *Euler matrix differential equation*. It is well known that regular solutions of scalar Euler differential equations ($n = 1$) can be written $y(x) = x^{\lambda_0}\, z(x)$ where $\lambda_0 \in \bar{\mathbb{K}}$ is a root of the indicial polynomial and $z(x) \in \bar{\mathbb{K}}[\log(x)]$ having a degree (in $\log(x)$) less than the multiplicity of λ_0 (see [8, Ch. 4]). We will show how this result can be extended to Euler matrix differential equations ($n \geq 2$).

To (3) we associate the matrix polynomial $L(\lambda)$ of size $n \times n$ and degree ℓ given by

$$L(\lambda) = A_\ell \lambda^\ell + \cdots + A_1\lambda + A_0. \qquad (4)$$

Its determinant will play the same role as the indicial polynomial in the scalar case.

2.1 Matrix polynomials

We recall some known properties of matrix polynomials which will be needed in the sequel to develop algorithms computing regular formal solutions of Systems (1) and (3). We refer to [11, Ch. 1] for more details.

We consider the matrix polynomial $L(\lambda)$ given by (4). Since A_ℓ is invertible, the determinant of $L(\lambda)$ is a polynomial in λ of degree $n\,\ell$. A root $\lambda_0 \in \bar{\mathbb{K}}$ of $\det(L(\lambda))$ is called *eigenvalue of* $L(\lambda)$ and its multiplicity is *the algebraic multiplicity of* λ_0 denoted by $m_a(\lambda_0)$. Let $\sigma(L)$ denote the set of all eigenvalues of $L(\lambda)$.

A non-zero vector $v \in \bar{\mathbb{K}}^n$ satisfying $L(\lambda_0)\, v = 0$ is called *eigenvector associated to* λ_0 and the dimension of $\ker(L(\lambda_0))$ is called the *geometric multiplicity of* λ_0 denoted by $m_g(\lambda_0)$.

Remark 1. In our algorithms, we need to compute a representation of the set $\sigma(L)$ of the eigenvalues of a matrix polynomial $L(\lambda)$. We proceed as follows: we compute and factor over $\mathbb{K}[\lambda]$ the determinant $\det(L(\lambda))$ of $L(\lambda)$. Then each eigenvalue $\lambda_0 \in \sigma(L)$ is represented by $\mathrm{RootOf}(\pi)$ where π is an irreducible factor of $\det(L(\lambda))$. Note that $\lambda_0 \in \mathbb{K}$ iff π is of degree 1. We can give an estimation of the arithmetic complexity for the calculation of $\sigma(L)$: computing the determinant can be done in $\mathcal{O}(n^\omega\, \ell)$ operations in $\mathbb{K}$ (see

[15]) where ω is the exponent of linear algebra. Factoring $\det(L(\lambda))$ over $\mathbb{K}$ can be done in $\mathcal{O}((n\,\ell)^{12})$ operations in $\mathbb{K}$ (see [19, Algorithm 18.7.3]). Note that, in the complexity analysis of Algorithms 1 and 2, we do not take into account the cost of the computation of $\sigma(L)$.

In what follows, for $p \in \mathbb{N}$, $L^{(p)}(\lambda)$ denotes the p-th derivative of $L(\lambda)$ w.r.t. λ.

Definition 1. Let $L(\lambda)$ be the matrix polynomial given by (4) and $\lambda_0 \in \sigma(L)$.

- A sequence of n-dimensional vectors $v_0 \neq 0, v_1, \dots, v_{k-1}$ satisfying
$$\sum_{p=0}^{i} \frac{L^{(p)}(\lambda_0)}{p\,!}\, v_{i-p} = 0, \quad \forall\, i = 0, \dots, k-1,$$
is called a *Jordan chain of length k associated to* λ_0. Note that v_0 is an eigenvector associated to λ_0 and $v_1, \dots, v_{k-1}$ are called *generalized eigenvectors associated to* λ_0.

- Let $v_0, v_1, \dots, v_{k-1}$ be a Jordan chain of length k associated to $\lambda_0 \in \sigma(L)$. The polynomial
$$\psi(\lambda) = \sum_{i=0}^{k-1} v_i\,(\lambda - \lambda_0)^i,$$
is called *root polynomial of $L(\lambda)$ of order k associated to* λ_0. It satisfies $L(\lambda_0)\,\psi(\lambda_0) = 0$ and the first $k-1$ derivatives of $L(\lambda)\psi(\lambda)$ vanish at $\lambda = \lambda_0$.

LEMMA 1. [11, Th. S1.10] *Let $L(\lambda)$ be a square matrix polynomial of size n of the form (4), $\lambda_0 \in \sigma(L)$ an eigenvalue of $L(\lambda)$ and $m_g = m_g(\lambda_0)$ its geometric multiplicity. Then, there exist two matrix polynomials $E_{\lambda_0}(\lambda)$ and $F_{\lambda_0}(\lambda)$, invertible at $\lambda = \lambda_0$, such that $L(\lambda) = E_{\lambda_0}(\lambda) S_{\lambda_0}(\lambda) F_{\lambda_0}(\lambda)$ where $S_{\lambda_0}(\lambda)$ is a diagonal matrix polynomial which diagonal entries are $1, \dots, 1, (\lambda-\lambda_0)^{\kappa_1}, \dots, (\lambda-\lambda_0)^{\kappa_{m_g}}$. The κ_i are positive integers satisfying $\kappa_1 \leq \dots \leq \kappa_{m_g}$. These integers are unique and called* the partial multiplicities associated to λ_0. *Moreover, we have $m_a(\lambda_0) = \sum_{i=1}^{m_g} \kappa_i$.*

Let us note that the partial multiplicities associated to a given eigenvalue λ_0 correspond to the maximum length of Jordan chains associated to λ_0. In [22, Ch. 3], Zúñiga proposes several algorithms for computing partial multiplicities and Jordan chains of maximal length associated to λ_0. As we need this auxiliary tool for our algorithms computing regular solutions of Systems (1) and (3), we have implemented Algorithm 3.3 in [22, Ch. 3] in Maple. The arithmetic complexity of this algorithm is at most $\mathcal{O}(n^5\,\ell^2\,d_{\lambda_0})$ operations in $\mathbb{K}$ where d_{λ_0} denotes the degree of the extension $\mathbb{K}(\lambda_0)$ over $\mathbb{K}$ (see [9, Ch. 1]).

Remark 2. Let $L(\lambda)$ be a square matrix polynomial of the form (4) and $\mathcal{C}$ the corresponding block companion matrix, then $\sigma(L(\lambda)) = \sigma(I\lambda - \mathcal{C})$ and the partial multiplicities associated to a given eigenvalue are the same for $L(\lambda)$ and $I\lambda - \mathcal{C}$. For more details, see [11, Ch. 1].

2.2 Euler matrix differential equation

>From [11, Ch. 1] and [22, Ch.3], we can derive the following algorithm which computes a basis of the regular solution space of (3).

ALGORITHM 1. **Input**: *the matrices A_i defining (3);* **Output**: *a basis of the regular solution space of (3);*

```
1. Let $L(\lambda)$ be given by (4); compute $\sigma(L)$;
2. For each eigenvalue $\lambda_0 \in \sigma(L)$:
    (a) Compute the partial multiplicities $\kappa_1, \dots, \kappa_{m_g(\lambda_0)}$
        associated to $\lambda_0$ and the Jordan chains of maximal
        length $v_{i,0}, \dots, v_{i,\kappa_i-1}$ for $i = 1, \dots, m_g(\lambda_0)$;
    (b) For $i = 1, \dots, m_g(\lambda_0)$ and $j = 0, \dots, \kappa_i - 1$, let
        $y_{\lambda_0,i,j}(x) = x^{\lambda_0} \sum_{k=0}^{j} v_{i,j-k} \frac{\log(x)^k}{k\,!}$;
3. Return all the $y_{\lambda_0,i,j}(x)$ obtained in Step 2.
```

PROPOSITION 1. *Algorithm 1 above is correct. For each $\lambda_0 \in \sigma(L)$, it uses at most $\mathcal{O}(n^5\,\ell^2\,d_{\lambda_0})$ operations in $\mathbb{K}$ where d_{λ_0} denotes the degree of the extension $\mathbb{K}(\lambda_0)$ over $\mathbb{K}$.*

PROOF. Let $\lambda_0 \in \sigma(L)$. From [11, Proposition 1.9], the system (3) has a solution of the form $x^{\lambda_0} \sum_{k=0}^{j} v_{i,j-k} \frac{\log(x)^k}{k\,!}$ with $v_{i,k} \in \bar{\mathbb{K}}^n$ and $v_{i,0} \neq 0$ iff $v_{i,0}, v_{i,1}, \dots, v_{i,j}$ form a Jordan chain of length j associated to λ_0. Moreover we know that the dimension of the regular solution space of (3) is equal to $n\,\ell$ (see [8, Ch. 4]). The correctness of the algorithm then follows from the linearly independence of the eigenvectors $v_{i,0}$, for $i = 1, \dots, m_g(\lambda_0)$ and the strictly ascending degree in $\log(x)$ of the polynomials $\sum_{k=0}^{j} v_{i,j-k} \frac{\log(x)^k}{k\,!}$ for $j = 0, \dots, \kappa_i - 1$, which imply the linearly independence of the solutions computed for the same λ_0. The complexity estimate follows directly from the fact that using Algorithm 3.3 in [22, Ch. 3], Step 2(a) can be performed using at most $\mathcal{O}(n^5\,\ell^2\,d_{\lambda_0})$ operations in $\mathbb{K}$ (see [9, Ch. 1]). □

2.3 Non-homogeneous systems

As we mentionned in the introduction, Poole's method for computing a basis of the regular formal solution space of (1) reduces to computing polynomial solutions in $t = \log(x)$ of non-homogeneous systems which coefficients are constant matrices and right-hand side is a vector of polynomials in t. The following proposition proves that such systems admit at least a polynomial solution in t and gives a bound on its degree.

PROPOSITION 2. *Let $t = \log(x)$ and consider the non-homogeneous system*
$$A_\ell \vartheta^\ell y + \dots + A_1 \vartheta y + A_0 y = \phi(t), \tag{5}$$
where for $i = 0, \dots, \ell$, $A_i \in \mathbb{K}^{n\times n}$, A_ℓ is invertible and $\phi(t)$ is a n-dimensional vector of polynomials in t of degree d. Let $L(\lambda)$ given by (4) be the associated matrix polynomial. Then, System (5) has at least a polynomial solution in t of degree p such that
$$\begin{array}{rl} d \leq p \leq d + \max\{\kappa_i, i = 1, \dots, m_g(0)\} & \text{if} \quad 0 \in \sigma(L), \\ p = d & \text{if} \quad 0 \notin \sigma(L), \end{array}$$
where $\kappa_1, \dots, \kappa_{m_g(0)}$ are the partial multiplicities of the eigenvalue 0 of $L(\lambda)$.

PROOF. This result can be proved directly using the properties of the matrix polynomial $L(\lambda)$ given by (4) but this is quite long and technical so we give another proof based on the associated non-homogeneous first-order system

$$\vartheta Y = \mathcal{C} Y + \Phi(t), \qquad (6)$$

where $\mathcal{C}$ is the block companion matrix (2) of size $n\,\ell$ with the $\tilde{A}_i = -A_\ell^{-1} A_i$ constant and $\Phi(t) = \begin{pmatrix} 0 & \cdots & 0 & \phi(t)^T \end{pmatrix}^T$. Let $Y = \sum_{i=0}^{p} Y_i \frac{t^i}{i!}$ with $Y_p \neq 0$ and $\Phi(t) = \sum_{i=0}^{d} \Phi_i \frac{t^i}{i!}$ with $\Phi_d \neq 0$. Since $\vartheta(\log(x)^i) = \frac{d}{dt}(t^i) = it^{i-1}$, the degree in t of $(\vartheta Y - \mathcal{C}Y)$ is always less or equal to the degree of Y. It follows that $d \leq p$. Suppose first that $\mathcal{C}$ is invertible, so, according to Remark 2, 0 is not an eigenvalue of $L(\lambda)$. Plugging $Y = \sum_{i=0}^{p} Y_i \frac{t^i}{i!}$ into (6), we get that the Y_i satisfy

$$\begin{cases} \mathcal{C}Y_p = -\Phi_p, \\ \mathcal{C}Y_i = Y_{i+1} - \Phi_i, \quad 0 \leq i \leq p-1, \end{cases}$$

assuming that $\Phi_i = 0$ for $i > d$. Since $\mathcal{C}$ is invertible, we necessarily have $p = d$ and the Y_i for $i = d, \ldots, 0$ can be computed recursively which provides a polynomial solution of (6) of degree $p = d$. Now we suppose that $\mathcal{C}$ is not invertible. Let P be an invertible constant matrix of size $n\,\ell$ such that the change of variables $Z = P\,Y$ transforms System (6) into $\vartheta Z = \mathcal{J} Z + \Psi(t)$, where $\mathcal{J}$ is the Jordan form of $\mathcal{C}$ and $\Psi(t) = P\,\Phi(t)$. Note that $\Psi(t)$ is a vector of polynomials of degree d in t. We can suppose without any loss of generality that $\mathcal{J} = \operatorname{diag}\{\mathcal{J}_1, \mathcal{J}_2\}$, where $\mathcal{J}_1$ is nilpotent and $\mathcal{J}_2$ is invertible. Let $Z = \begin{pmatrix} Z_1^T & Z_2^T \end{pmatrix}^T$ and $\Psi(t) = \begin{pmatrix} \Psi_1(t)^T & \Psi_2(t)^T \end{pmatrix}^T$ be decomposed into blocks according to the decomposition of $\mathcal{J}$. System (6) is then equivalent to

$$\vartheta Z_i = \mathcal{J}_i Z_i + \Psi_i(t), \quad i = 1, 2. \qquad (7)$$

Since $\mathcal{J}_2$ is invertible, System (7) for $i = 2$ has a polynomial solution in t of degree at most d. Now for $i = 1$, the problem can be reduced to that of finding polynomial solutions of the first-order system of size κ

$$\vartheta X = J\,X + F(t), \qquad (8)$$

where κ is a partial multiplicity of the eigenvalue 0 of $L(\lambda)$, J is a nilpotent Jordan block of the form

$$J = \begin{pmatrix} 0 & 1 & \cdots & 0 \\ & \ddots & \ddots & \vdots \\ & (0) & \ddots & 1 \\ & & & 0 \end{pmatrix}$$

and $F(t)$ is a vector of polynomials of degree bounded by d. Let $X = \begin{pmatrix} X_1 & \cdots & X_\kappa \end{pmatrix}^T$ and $F(t) = \begin{pmatrix} F_1(t) & \cdots & F_\kappa(t) \end{pmatrix}^T$. Then System (8) is equivalent to

$$\begin{cases} \vartheta X_\kappa = F_\kappa(t), \\ \vartheta X_i = X_{i+1} + F_i(t), \quad 0 \leq i \leq \kappa - 1. \end{cases}$$

Since $F_\kappa(t)$ is a polynomial in t, X_κ can be obtained by integration and is also a polynomial in t. We can then compute recursively the polynomials X_i for $i = \kappa-1, \ldots, 0$ by integration. Hence, the degree in t of the polynomial solution X of (8) that we have obtained is less or equal to $\kappa + d$. □

3. THE GENERAL CASE

In this section, we develop algorithms computing a basis of the regular formal solution space of System (1) with variable coefficients. For $i = 0, \ldots, \ell$, let $A_i(x) = \sum_{j=0}^{\infty} A_{ij} x^j$ where $A_{ij} \in \mathbb{K}^{n\times n}$ and $A_{\ell 0} = A_\ell(0)$ is invertible. Our system (1) can be written as

$$\mathcal{L}(y(x)) = \sum_{j=0}^{\infty} x^j L_j(\vartheta) y(x) = 0, \qquad (9)$$

where the $L_j(\lambda)$ are the matrix polynomials defined by

$$L_j(\lambda) = A_{\ell j}\lambda^\ell + \cdots + A_{1j}\lambda + A_{0j}, \quad j \geq 0. \qquad (10)$$

3.1 Generalization of Poole's method

In this subsection, we prove that Poole's approach for computing regular solutions of a scalar linear differential equation ([20, Chap. 5, § 16]) can be extended to handle the general system (1). This method consists in computing regular solutions viewed as power series which coefficients are vectors of polynomials in $t = \log(x)$, *i.e.*, of the form

$$y(x) = x^{\lambda_0} \sum_{i \geq 0} U_i(t)\, x^i, \qquad (11)$$

where $\lambda_0 \in \bar{\mathbb{K}}$, for all $i \geq 0$, $U_i(t)$ is a n-dimensional vector of polynomials in t of degree less than $n\,\ell$ and $U_0 \neq 0$.

Plugging (11) into (9) and using the equality

$$L_j(\vartheta)\left(x^{\lambda_0+i} U_i\right) = x^{\lambda_0+i} L_j(\vartheta + \lambda_0 + i) U_i,$$

we find that U_0 satisfies

$$L_0(\vartheta + \lambda_0) U_0 = 0, \qquad (12)$$

and for $m \geq 1$, U_m satisfies

$$L_0(\vartheta + \lambda_0 + m)\, U_m = -\sum_{i=0}^{m-1} L_{m-i}(\vartheta + \lambda_0 + i)\, U_i. \qquad (13)$$

The problem is then reduced to finding $U_0, U_1, \ldots, \in \bar{K}[t]^n$ that satisfy (12) and (13).

Equation (12) implies $x^{\lambda_0} L_0(\vartheta + \lambda_0)\, U_0 = 0$ which is equivalent to $L_0(\vartheta)\left(x^{\lambda_0} U_0\right) = 0$. Now, $L_0(\vartheta)(y) = 0$ is a Euler matrix differential equation hence it can be solved by Algorithm 1. It then follows that λ_0 is an eigenvalue of $L_0(\lambda)$ and U_0 is of the form

$$U_0 = \sum_{i=0}^{k-1} v_{k-1-i} \frac{\log(x)^i}{i\,!}, \qquad (14)$$

where $v_0, v_1, \ldots, v_{k-1}$ is a Jordan chain of length k associated to λ_0.

Let $\lambda_0 \in \sigma(L_0)$. For each choice of U_0 of the form (14) with $k = \kappa(\lambda_0)$ a partial multiplicity of λ_0, we first consider Equation (13) for $m = 1$. This is a non-homogeneous system with constant coefficients which right-hand side is a vector of polynomials in t of degree $d \leq \kappa(\lambda_0)$. From Proposition 2, we know that it admits at least a solution U_1 polynomial in t. Its degree is bounded by $\kappa(\lambda_0)$ if $\lambda_0 + 1$ is not an eigenvalue of L_0 and is bounded by $\kappa(\lambda_0) + \max_j\{\kappa_j(\lambda_0 + 1)\}$, where the $\kappa_j(\lambda_0 + 1)$ are the partial multiplicities of the eigenvalue $\lambda_0 + 1$, otherwise. Recursively, we can then obtain $U_2, U_3, \ldots \in \bar{K}[t]^n$ as polynomial solutions of the non-homogeneous systems (13) for $m = 2, 3, \ldots$. It follows that the degree of U_m is bounded by

$$\kappa(\lambda_0) + \sum_{1\le i\le m \,\mid\, \lambda_0+i\in\sigma(L_0)} \max_j\{\kappa_j(\lambda_0+i)\} \qquad (15)$$

where the $\kappa_j(\lambda_0+i)$ are the partial multiplicities of the eigenvalue λ_0+i.

We shall now explain how to compute efficiently a polynomial solution U_m of (13). We suppose thus that $\lambda_0 \in \sigma(L_0)$ and $U_0 = \sum_{i=0}^{\kappa(\lambda_0)-1} v_{\kappa(\lambda_0)-1-i}\frac{\log(x)^i}{i\,!}$ are fixed and that $U_1, \ldots, U_{m-1}$ have been computed. Let $\rho = \lambda_0 + m$, $U_m = \sum_{i=0}^{p} U_{mi}\,\frac{t^i}{i!}$ where p is given by (15) and the U_{mi} are constant vectors to be determined. Write the right-hand side of (13) under the form $\sum_{i=0}^{d} q_i\,\frac{t^i}{i!}$, where $q_d \neq 0$.

Writing $L_0(\vartheta+\rho) = \sum_{i=0}^{\ell}\frac{L_0^{(i)}(\rho)}{i!}\vartheta^i$, we see that System (13) is equivalent to the linear system of size $(p+1)\,n$

$$\begin{pmatrix} L_0(\rho) & & & \\ \vdots & \ddots & (0) & \\ \frac{L_0^{(p-d)}(\rho)}{(p-d)!} & & \ddots & \\ \vdots & & & \ddots \\ \frac{L_0^{(p)}(\rho)}{p!} & \ldots & \ldots & \ldots & L_0(\rho)\end{pmatrix}\begin{pmatrix}U_{mp}\\ \vdots \\ U_{md} \\ \vdots \\ U_{m0}\end{pmatrix} = \begin{pmatrix}0\\ \vdots\\ 0\\ q_d\\ \vdots\\ q_0\end{pmatrix},$$

which is equivalent to the $p+1$ linear systems of size n

$$L_0(\rho)\,U_{mi} = f_i, \quad 0 \le i \le p, \qquad (16)$$

where $f_p = q_p$, $f_i = q_i - \sum_{j=1}^{p-i}\frac{L_0^{(j)}(\rho)}{j!}U_{m(i+j)}$ for $0 \le i \le p-1$, and $q_i = 0$ for $i > d$. Note that all the systems in (16) share the same matrix $L_0(\rho)$.

The idea is to first compute a LU decomposition of $L_0(\rho)$ and then use it to compute recursively the U_{mi}: let $L_0(\rho) = P\,L\,U$ where P is a permutation matrix, L a lower triangular matrix having 1 on its diagonal and U is an upper triangular matrix which last $n - \mathrm{rank}(L_0(\rho))$ rows are zero. To obtain a solution of the first system $L_0(\rho)U_{mp} = q_p$, we solve $L\,z = P^{-1}\,q_p$. The solution z depends on arbitrary constants. Then we consider the system $U\,U_{mp} = z$: as it has to be consistent, the values of some of the latter constants are fixed so that the last $n-\mathrm{rank}(L_0(\rho))$ components of z are zero. This system is then solved and we get U_{mp}. The U_{mi} for $i = p-1, \ldots, 0$ are computed recursively in the same way.

Remark 3. A significant number of systems (13) have to be considered (one for each couple (λ_0, m)) whereas $\rho = \lambda_0 + m$ is an eigenvalue of $L_0(\lambda)$ for only a few of them. Consequently, in our implementation, we first compute formally the inverse of the matrix $L_0(\lambda)$. Then, if ρ is not an eigenvalue of $L_0(\lambda)$, we solve the systems (16) by evaluating the inverse at ρ and computing $U_{mi} = L_0^{-1}(\rho)\,f_i$, otherwise we compute a LU decomposition of $L_0(\rho)$ and proceed as explained above.

ALGORITHM 2. **Input***: the A_i defining (1) and $\nu \in \mathbb{N}^*$;* **Output***: a basis of the regular solution space of (1) where the involved power series are computed up to order ν;*

```
1. Let $L_0(\lambda)$ be given by (10). Compute $\sigma(L_0)$;
2. For each eigenvalue $\lambda_0 \in \sigma(L_0)$:
    (a) Compute the partial multiplicities $\kappa_1, \ldots, \kappa_{m_g(\lambda_0)}$
        associated to $\lambda_0$ and the Jordan chains of maximal
        length $v_{i,0}, \ldots, v_{i,\kappa_i-1}$ for $i = 1, \ldots, m_g(\lambda_0)$;
    (b) For $i = 1, \ldots, m_g(\lambda_0)$ and $j = 0, \ldots, \kappa_i - 1$ :
        i. Let $U_{(i,j),0} = \sum_{k=0}^{j} v_{i,j-k}\frac{\log(x)^k}{k\,!}$;
        ii. For $m = 1, \ldots, \nu$, compute $U_{(i,j),m}$ from Equation (13) (as explained above);
3. Return all the $y_{\lambda_0,i,j}(x) = x^{\lambda_0}\sum_{m=0}^{\nu} U_{(i,j),m}\,x^m$ with
   the $U_{(i,j),m}$ computed in Step 2.
```

Remark 4. In Step 2(b), for each $i = 1, \ldots, m_g(\lambda_0)$, it is sufficient to compute the $U_{(i,j),m}$ for $j = \kappa_i - 1$: indeed the $U_{(i,j),m}$ for $j = 0, \ldots, \kappa_i - 2$ can be easily deduced from those. If eigenvalues of L_0 differ by integers, we proceed with them by decreasing order: hence the bound (15) needed in Step 2(b)ii is always known without any extra computation.

PROPOSITION 3. *Algorithm 2 above is correct. For each $\lambda_0 \in \sigma(L_0)$, it uses at most $\mathcal{O}(n^5\,\ell^3\,\nu^2\,d_{\lambda_0})$ operations in $\mathbb{K}$ where d_{λ_0} denotes the degree of the extension $\mathbb{K}(\lambda_0)$ over $\mathbb{K}$.*

PROOF. The correctness of the algorithm follows from the explanations above, the linearly independence of the vectors $v_{i,0}$, for $i = 1, \ldots, m_g(\lambda_0)$, and the strictly ascending degree in $\log(x)$ of the $U_{(i,j),0}$, for $j = 0, \ldots, \kappa_i - 1$, which imply the linearly independence of the solutions computed for the same λ_0. We have already seen that Step 2(a) can be performed using $\mathcal{O}(n^5\,\ell^2\,d_{\lambda_0})$ operations in $\mathbb{K}$. Now we have to give an estimation for the cost of computing a solution $U_m = U_{(i,j),m}$ of (13). First we have to compute the right-hand sides f_i of System (16). To achieve this, we need to compute the $L_0^{(j)}(\lambda)$ for $j = 0, \ldots, p$: this reduces to differentiate n^2 polynomials of degree $\le \ell$ at most ℓ times. Each differentiation costs $\mathcal{O}(\ell)$ arithmetic operations so that the total cost for computing the $L_0^{(j)}(\lambda)$ for $j = 0, \ldots, p$ is bounded by $\mathcal{O}(n^2\,\ell^2)$ operations in $\mathbb{K}$. Then we have to evaluate each of those at ρ. This is equivalent to evaluate at most $n^2\,\ell$ polynomials of degree bounded by ℓ at $\rho = \lambda_0 + m$. Evaluating a polynomial of degree $\le \ell$ with coefficients in $\mathbb{K}$ at $\rho \in \mathbb{K}(\lambda_0)$ can be done using at most $\mathcal{O}(\ell\,d_{\lambda_0})$ operations in $\mathbb{K}$ (see [7]). Hence the cost of the evaluation is $\mathcal{O}(n^2\,\ell^2\,d_{\lambda_0})$ operations in $\mathbb{K}$. Then to compute a right-hand side f_i of the system (16), we need at most ℓ products matrix-vector of size n which costs at most $\mathcal{O}(n^2\,\ell\,d_{\lambda_0})$ operations in $\mathbb{K}$. As there are $p+1$ systems and $p \le n\,\ell$, the total cost of these multiplications is thus $\mathcal{O}(n^3\,\ell^2\,d_{\lambda_0})$ operations in $\mathbb{K}$. With the same tools, we obtain that all the q_i can be computed using at most $\mathcal{O}(n^3\,\ell^2\,m\,d_{\lambda_0})$ operations in $\mathbb{K}$ so that, finally, the cost of the computation of the f_i is bounded by $\mathcal{O}(n^3\,\ell^2\,m\,d_{\lambda_0})$ operations in $\mathbb{K}$. Then we have to solve the $p+1$ systems of (16): the complexity of solving all these systems is bounded by $\mathcal{O}(n^4\,\ell\,d_{\lambda_0})$. Consequently, the cost of computing a $U_{(i,j),m}$ in Step 2(b)ii is bounded by $\mathcal{O}(n^4\,\ell^2\,m\,d_{\lambda_0})$ operations in $\mathbb{K}$ and that of computing all the $U_{(i,j),m}$ for $m = 1, \ldots, \nu$ is bounded by $\mathcal{O}(n^4\,\ell^2\,\nu^2\,d_{\lambda_0})$. As $m_g(\lambda_0) \le n\,\ell$ and taking into account Remark 4, we obtain the result of the algorithm. $\square$

3.2 Generalization of Frobenius' method

A classical method for computing a basis of the regular solution space of a scalar linear differential equation is that of Frobenius (see [10], [14, Ch. 16] or [8, Ch. 4, Section 8]). This method has been extended to the case of linear differential systems of first-order (see [8, Ch. 4] for example). In this subsection, we briefly explain how it can be generalized to handle general linear differential systems of the form (1).

The idea consists in computing regular formal solutions of the form $y(\lambda_0, x) = x^{\lambda_0} \sum_{i\geq 0} g_i(\lambda_0)\, x^i$, where λ_0 is an eigenvalue of the matrix polynomial $L_0(\lambda)$ defined by (10). Here, $\det(L_0(\lambda))$ replaces the indicial polynomial in the scalar case. To each eigenvalue $\lambda_0 \in \sigma(L_0)$, we associate $m_a(\lambda_0)$ linearly independent solutions. As in the scalar and first-order cases, we distinguish two cases:

3.2.1 *No eigenvalues differing by an integer*

Let λ_0 be an eigenvalue of $L_0(\lambda)$ and $\kappa_1, \ldots, \kappa_{m_g(\lambda_0)}$ its partial multiplicities. We suppose here that $\forall i \geq 1$, $\lambda_0 + i$ is not an eigenvalue of $L_0(\lambda)$, *i.e.*, $L_0(\lambda_0 + i)$ is invertible.

We consider the non-homogeneous system

$$\mathcal{L}(y) = L_0(\lambda)\, g_0(\lambda)\, x^\lambda, \tag{17}$$

where $g_0(\lambda)$ is an arbitrary n-dimensional vector depending on λ and we look for logarithm-free regular formal solutions of (17) of the form

$$y(\lambda, x, g_0) = x^\lambda \sum_{i\geq 0} g_i(\lambda) x^i, \tag{18}$$

where, for $i \geq 1$, $g_i(\lambda)$ is a n-dimensional vector function of λ to be determined.

A direct computation shows that plugging (18) into (17) yields

$$L_0(\lambda + i)\, g_i(\lambda) = -\sum_{j=1}^{i} L_j(\lambda + i - j)\, g_{i-j}(\lambda), \quad i \geq 1. \tag{19}$$

Since for $i \geq 1$, $L_0(\lambda_0 + i)$ is invertible, $g_i(\lambda)$ is uniquely determined as a rational function of λ well defined in a neighborhood of λ_0. Then, we have the following: if $g_0(\lambda_0)$ is an eigenvector of $L_0(\lambda)$ associated to λ_0, then

$$y(\lambda_0, x, g_0) = x^{\lambda_0} \sum_{i\geq 0} g_i(\lambda_0)\, x^i,$$

where, for $i \geq 1$, the $g_i(\lambda)$ are uniquely defined by (19), is a solution of (1).

If $m_a(\lambda_0) = 1$, then we are done since we have computed a (logarithm-free) regular solution associated to λ_0. Otherwise, as in the scalar and first-order cases, the other solutions involving logarithm-terms are obtained by differentiations w.r.t. λ. We have $m_a(\lambda_0) = \sum_{i=1}^{m_g(\lambda_0)} \kappa_i$. For $i = 1, \ldots, m_g(\lambda_0)$, we choose $g_{i,0}(\lambda)$ as a root polynomial of maximal order κ_i associated to λ_0 and we compute the $\frac{\partial^j y}{\partial \lambda^j}(\lambda_0, x, g_{i,0}), j = 0, \ldots, \kappa_i - 1$ where $y(\lambda, x, g_{i,0})$ is the logarithm-free solution of (17) computed for $g_0(\lambda) = g_{i,0}(\lambda)$ as explained above. It yields $m_a(\lambda_0)$ linearly independent solutions of (1) associated to λ_0.

3.2.2 *Eigenvalues differing by integers*

We suppose now that there exist exactly r eigenvalues $\lambda_1, \ldots, \lambda_r$ of $L_0(\lambda)$ such that $\Re(\lambda_1) < \cdots < \Re(\lambda_r)$ where $\Re(z)$ denotes the real part of a complex number $z \in \mathbb{C}$, and for $j > i$, $\lambda_j - \lambda_i \in \mathbb{N}^*$. For $i \in \{1, \ldots, r-1\}$ let $\alpha_i = \sum_{p=i+1}^{r} m_a(\lambda_p)$. The solutions associated to λ_r can be determined as in the first case studied in the previous subsection but those associated to λ_i, for $i \neq r$, are determined in a slightly different way.

Instead of (17), we consider the non-homogeneous system

$$\mathcal{L}(y) = (\lambda - \lambda_i)^{\alpha_i} L_0(\lambda) g_0(\lambda) x^\lambda, \tag{20}$$

where $g_0(\lambda)$ is an arbitrary n-dimensional vector depending on λ and we look for logarithm-free regular formal solutions of (17) of the form

$$y(\lambda, x, g_0) = \left((\lambda - \lambda_i)^{\alpha_i} g_0(\lambda) + \sum_{k\geq 1} g_k(\lambda) x^k \right) x^\lambda, \tag{21}$$

where, for $i \geq 1$, $g_i(\lambda)$ is a n-dimensional vector function of λ to be determined.

A direct computation shows that, plugging (21) into (20), we obtain

$$\begin{aligned} L_0(\lambda + k) g_k(\lambda) = &-\sum_{j=1}^{k-1} L_j(\lambda + k - j) g_{k-j}(\lambda) \\ &- (\lambda - \lambda_i)^{\alpha_i} L_k(\lambda) g_0(\lambda), \quad k \geq 1. \end{aligned} \tag{22}$$

When $\lambda_i + k$ is not an eigenvalue of $L_0(\lambda)$, $g_k(\lambda)$ is uniquely determined as a vector of rational functions of λ defined in a neighborhood of λ_i. Now, if $\lambda_i + k$ is an eigenvalue of $L_0(\lambda)$, we can also get a solution $g_k(\lambda)$ of (22) well defined at λ_i. Indeed, the presence of the term $(\lambda - \lambda_i)^{\alpha_i}$ in (20) and (21) implies that the right-hand side of (22) admits $(\lambda - \lambda_i)$ as a factor of multiplicity greater or equal to the multiplicity of $(\lambda - \lambda_i)$ in the determinant of $L_0(\lambda + k)$. So the simplification can be performed and we obtain a $g_k(\lambda)$ defined in a neighborhood of λ_i. In this way, we can solve all the systems (22) and compute a (logarithm-free) regular solution of (20) by evaluating (21) at $\lambda = \lambda_i$. However, since we search for solutions that are linearly independent from those associated to λ_r, we proceed as follows (note that the analog technique is used in the scalar and first-order cases): we have $m_a(\lambda_i) = \sum_{j=1}^{m_g(\lambda_i)} \kappa_j$ where $\kappa_1, \ldots, \kappa_{m_g(\lambda_i)}$ denote the partial multiplicities of λ_i. For $j = 1, \ldots, m_g(\lambda_i)$, we choose $g_{j,0}(\lambda)$ as a root polynomial associated to λ_i of maximal order κ_j and we compute the $\frac{\partial^k y}{\partial \lambda^k}(\lambda_i, x, g_{j,0}), k = \alpha_i, \ldots, \alpha_i + \kappa_j - 1$ where $y(\lambda, x, g_{j,0})$ is the logarithm-free regular solution of (20) computed for $g_0(\lambda) = g_{j,0}(\lambda)$ as explained above. It provides κ_j linearly independent solutions associated to λ_i.

4. AN EXTENSION

In this section, we consider a system of the form

$$\mathcal{L}(y(x)) = \sum_{i=0}^{\ell} A_i(x) \vartheta^i y(x) = 0, \tag{23}$$

with $A_\ell(x)$ invertible in $\mathbb{K}((x))^{n\times n}$. Here, $A_\ell(0)$ is no longer assumed to be invertible and may be equal to zero. Let $A_i(x) = \sum_{j=0}^{\infty} A_{ij}\, x^j$ and

$$L_0(\lambda) = A_{\ell 0}\lambda^\ell + \cdots + A_{10}\lambda + A_{00}. \tag{24}$$

We further suppose that $\det(L_0(\lambda))$ is not identically zero so that the two methods developed in the previous section (in particular Algorithm 2) can be used to compute $\deg(\det(L_0(\lambda)))$ linearly independent regular formal solutions of System (23). The following proposition shows that this is exactly the dimension of the regular solution space of (23).

Proposition 4. *Let $L_0(\lambda)$ be given by (24). If $\det(L_0(\lambda))$ is not identically zero, then the dimension of the regular formal solution space of (23) is equal to $\deg(\det(L_0(\lambda)))$.*

Proof. System (23) is equivalent to the first-order system $D(x)\,\vartheta\,Y(x) = N(x)\,Y(x)$ where $D(x)$ is the block diagonal matrix $\mathrm{diag}\{I, \dots, I, A_\ell(x)\}$ and $N(x)$ is the block companion matrix of the form (2) with last row equal to $[\ -A_0(x) \quad -A_1(x) \quad \dots \quad -A_{\ell-1}(x)\ ]$. Since the two matrices $D(x)$ and $N(x)$ are analytic at the origin, $D(x)$ is invertible and $\det(D(0)\lambda - N(0)) = \det(L_0(\lambda))$ (see [11, Ch.7]) is not identically zero, the resulting first-order system is simple in the sense of [5]. Hence, applying [5, Theorem 3.2], the dimension of the regular solution space of (23) is equal to $\deg(\det(D(0)\lambda - N(0))) = \deg(\det(L_0(\lambda)))$. $\square$

With the hypotheses of Proposition 4, System (23) has an irregular singularity at the origin iff $\deg(\det(L_0(\lambda))) < n\,\ell$, *i.e.*, iff $A_{\ell 0}$ is singular (see [11, Ch. 7]).

Table 1: Timings (in seconds) for $\ell = 1$

$\boldsymbol{n}$	$\boldsymbol{\sigma(L_0)}$	Bce	Lfs	Fos
2	$\{1, 2\}$	0.209	0.060	0.108
4	$\{1, 2\}$	0.785	0.177	0.236
8	$\{1, 2, 3\}$	7.098	1.096	1.045
16	$\{1, 2, 3, 4\}$	78.121	24.725	21.263
2	$\{\alpha_0\}$	0.221	0.216	0.164
4	$\{\alpha_0, 0, 1\}$	0.821	0.994	0.480
8	$\{\alpha_0, 0, \dots, 5\}$	7.706	6.730	2.810
12	$\{\alpha_0, 0, \dots, 9\}$	50.721	54.234	73.525
2	$\{1, \frac{1}{2}\}$	0.110	0.081	0.165
4	$\{1, \frac{1}{2} \dots, \frac{1}{4}\}$	0.266	0.429	0.584
8	$\{1, \frac{1}{2}, \dots, \frac{1}{8}\}$	0.920	3.630	3.669
16	$\{1, \frac{1}{2} \dots, \frac{1}{16}\}$	20.652	88.333	133.377
2	$\{1\}$	0.119	0.055	0.124
4	$\{1\}$	0.264	0.139	0.255
8	$\{1\}$	0.878	0.692	0.743
16	$\{1\}$	8.411	7.121	7.977
2	$\{\alpha_1\}$	0.229	0.218	0.169
4	$\{\alpha_1, \alpha_2\}$	1.875	1.666	0.555
6	$\{\alpha_1, \alpha_2, \alpha_3\}$	7.079	5.223	1.711

5. IMPLEMENTATION AND TIMINGS

We have implemented both Algorithm 2 and the generalization of Frobenius' method in Maple using the package LinearAlgebra for the linear algebra manipulations and MatrixPolynomialAlgebra for the matrix polynomial manipulations. They compute a basis of the regular solution space of a system of the form (23), where the involved power series of the solutions are computed up to a fixed order ν.

Our prototype of the Frobenius approach follows exactly the method sketched in Subsection 3.2. When the associated matrix polynomial has eigenvalues with too large partial multiplicities, a lot of differentiations and evaluations have to be performed in order to compute all regular solutions. This may lead to the saturation of the memory machine. As a consequence, our implementation of Frobenius' method turned out to work only for systems of small size and order. Consequently, we will not give any timings of it in this paper but we plan to work on an improved implementation by considering the linear systems hidden behind all these differentiations and evaluations.

In the following, we give some timings[2] comparing three implementations:

1. Bce[3]: implements Algorithm 2;
2. Lfs: implements the algorithm of [3] available in the package LinearFunctionalSystems;
3. Fos: convert (23) into a first-order system of size $n\,\ell$ and then use the implementation available in the package Isolde (see [6]) of the algorithm given in [5].

First, we consider first-order systems of size n of the form $\vartheta y + A(x)y = 0$, where $A(x)$ is a matrix with polynomial coefficients. Note that this case is of interest since, to our knowledge, Poole's approach has not been generalized for first-order linear differential system before. The associated matrix polynomial is $L_0(\lambda) = I\lambda + A(0)$, where I denotes the identity matrix of size n. The systems are constructed as follows: we choose a set $\mathcal{S}$ of n elements in $\bar{\mathbb{Q}}$ not necessarily distinct. Then, we construct a matrix $A(0) \in \mathbb{Q}^{n\times n}$, such that $\sigma(L_0) = \mathcal{S}$. Finally, using random matrices, we generate 20 matrix $A(x) \in \mathbb{Q}[x]^{n\times n}$ of degree 8 and we run the three algorithms. Timings are presented in Table 1: they indicate the average time in seconds taken for computing a basis of the regular solution space. The power series are computed up to order $\nu = 5$ and we note $\alpha_0 = \mathrm{RootOf}(\_Z^2 + 1)$, $\alpha_1 = \mathrm{RootOf}(\_Z^2 - 2\_Z + 2)$, $\alpha_2 = \mathrm{RootOf}(\_Z^2 - 4\_Z + 5)$ and $\alpha_3 = \mathrm{RootOf}(\_Z^2 - 6\_Z + 10)$. We can re! mark that, on these examples, our algorithm seems only faster than the others when the eigenvalues are all distinct and do not differ by integers. Otherwise, Lfs and Fos are a little bit faster. However, the performances of Bce, which is not designed for first-order systems, are quite correct.

Note that an advantage of our implementation (and of Fos) is that it can handle systems having coefficients in an extension $\mathbb{K}$ of $\mathbb{Q}$, *i.e.*, $A(x) \in \mathbb{K}[[x]]^{n\times n}$ which is not the case for Lfs. For example, consider $\mathbb{K} = \mathbb{Q}(i)$, where $i^2 + 1 = 0$ and choose $L_0(\lambda) \in \mathbb{K}[\lambda]^{n\times n}$, such that $\sigma(L_0) \subseteq \{\alpha_1, \alpha_2, \alpha_3\}$. If we generate 20 random systems as we did before, the experimental results show that Bce takes on average 0.296 seconds (resp. 3.931) for first-order systems of size $n = 2$ (resp. $n = 4$) whereas Fos takes on average 0.428 seconds (resp. 22.977).

Then, we compare timings for higher-order ($\ell \geq 2$) linear differential systems of the form (1). We construct such systems as follows: choose a set $\mathcal{S}$ of n rational numbers not necessarily distinct and a diagonal matrix polynomial $D(\lambda)$ s.t. the degrees of its diagonal entries are exactly equal to ℓ and their roots are among the elements of $\mathcal{S}$, then take $L(\lambda) = PD(\lambda)P^{-1}$, where P is an invertible random ma-

[2]Computations were made on a 2.4 GHz Intel Core 2 Duo
[3]available at `http://www.ensil.unilim.fr/~cluzeau`

Table 2: Timings (in seconds) for $n = 2$

$\boldsymbol{\ell}$	$\boldsymbol{\sigma(L_0)}$	Bce	Lfs	Fos
4	$\{\frac{1}{2}, \frac{5}{2}, \frac{4}{3}\}$	0.862	0.677	0.858
8	$\{\frac{1}{2}, \frac{5}{2}, \frac{4}{3}\}$	2.346	6.229	4.671
12	$\{\frac{1}{2}, \frac{5}{2}, \frac{4}{3}\}$	5.619	28.807	21.714
4	$\{\frac{1}{2}, 1\}$	0.498	0.626	1.151
8	$\{\frac{1}{2}, 1\}$	1.490	5.301	5.135
12	$\{\frac{1}{2}, 1\}$	3.625	24.195	19.315

Table 3: Timings (in seconds) for $\ell = 4$

$\boldsymbol{n}$	$\boldsymbol{\sigma(L_0)}$	Bce	Lfs	Fos
2	$\{\frac{1}{2}, \frac{5}{2}, \frac{4}{3}\}$	0.926	0.865	1.083
5	$\{\frac{1}{2}, \frac{5}{2}, \frac{4}{3}\}$	8.263	17.359	24.336
8	$\{\frac{1}{2}, \frac{5}{2}, \frac{4}{3}\}$	25.075	95.329	308.903
2	$\{\frac{1}{2}, 1\}$	0.462	0.644	1.130
5	$\{\frac{1}{2}, 1\}$	2.121	10.774	20.228
8	$\{\frac{1}{2}, 1\}$	7.282	60.397	246.695

trix. Finally, generate 20 systems of the form (1) where the degree of the entries of the matrices $A_i(x)$ is equal to 6 and its associated matrix polynomial $L_0(\lambda)$ is $L(\lambda)$. Timings are given in Table 2, where n is fixed to 2, and in Table 3, where ℓ is fixed to 4. In both tables, the power series are computed up to order $\nu = 5$. These timings seems to show that Bce is almost always faster than Lfs and Fos (at least when ν is small). This can be e! xplained by the fact that our program uses a direct approach while Lfs uses the desingularization of the associated matrix recurrence (see [1, 2]) and Fos manipulates a first-order system of bigger size $n\,\ell$ without taking advantage of its companion structure. We have also made tests for fixed values of n and ℓ and increasing values of ν. The timings obtained confirm the efficiency of bce for small values of ν. For larger values of ν, our tests do not allow to draw any conclusion so that these experiences deserve to be continued.

Acknowledgments

The authors are grateful to anonymous referees for their valuable comments and remarks which have improved the clarity of the article.

6. REFERENCES

[1] S. Abramov. EG - eliminations. *Journal of Difference Equations and Applications*, 5:393-433, 1999.

[2] S. Abramov and M. Bronstein. On Solutions of Linear Functional Systems. In *Proceedings of ISSAC '2001*, ACM Press.

[3] S. Abramov, M. Bronstein and D. E. Khmelnov. On Regular and Logarithmic Solutions of Ordinary Linear Differential Systems. *Computer Algebra in Scientific Computing, Lecture Notes in Computer Science, Springer Verlag*, 3718:1-12, 2005.

[4] W. Balser. Formal Power Series and Linear Systems of Meromorphic Ordinary Differential Equations. *Springer*, 2000.

[5] M. Barkatou and E. Pflügel. An Algorithm Computing the Regular Formal Solutions of a System of Linear Differential Equations. *J. Symbolic Computation*, 11:1-20, 1998.

[6] M. Barkatou and E. Pflügel. The ISOLDE package. A SourceForge Open Source project. `http://isolde.sourceforge.net`, 2006.

[7] A. Bostan. Algorithmique efficace pour des opérations de base en Calcul Formel. *Thèse de l'école Polytechnique*, 2003.

[8] E. Coddington and N. Levinson. Theory of Ordinary Differential Equations, *McGraw-Hill Book Company, Inc.,* 1955.

[9] C. El Bacha. Matrices polynomiales et applications à la résolution symbolique des systèmes d'équations différentielles d'ordre arbitraire. Master 2 report, University of Limoges, 2008.

[10] G. Frobenius. Über die Integration der Linearen Differentialgleichungen mit Veränder Koefficienten. *Journal für die reine und angewandte Mathematik*, 76:214-235, 1873.

[11] I. Gohberg, P. Lancaster and L. Matrix Polynomials. *Academic Press*, New York, 1982.

[12] L. Heffter. Einleitung in die Theorie der Linearen Differentialgleichungen. *Teubner*, Leipzig, 1894.

[13] M. Humi. Factorization of Systems Differential Equations. *J. Math. Phys.*, 27(1):76-81, 1986.

[14] E. L. Ince. Ordinary Differential Equations. *Dover Publications Inc.*, 1926.

[15] C. P. Jeannerod and G. Villard. Asymptotically Fast Polynomial Matrix Algorithms for Multivariable Systems. *International Journal of Control*, 79(11):1359-1367, 2006.

[16] L. Jódar and M. Legua. Solving Second-Order Matrix Differential Equations with Regular Singular Points Avoiding the Increase of the Problem Dimension. *Applied Mathematics and Computation*, 53:191-205, 1993.

[17] E. Navarro, L. Jódar and R. Company. Solving Higher Order Fuchs Type Differential Systems Avoiding the Increase of the Problem Dimension. *International Journal of Mathematics and Mathematics Sciences*, 17(1):91-102, 1994.

[18] D. L. Littlefield and P. V. Desai. Frobenius Analysis of Higher Order Equations: Incipient Buoyant Thermal Convection. *SIAM J. Appl. Math.*, 50(6):1752-1763, 1990.

[19] T. Mora. Solving Polynomial Systems I : The Kronecker-Duval Philosophy. *Encyclopedia of Mathematics and its Applications 88,* Cambridge University Press, 2003.

[20] E. G. C. Poole. Introduction to the Theory of Linear Differential Equations. *Oxford University Press*, LONDON, 1936.

[21] Y. Gong Sun and F. Wei Meng. Oscillation Results for Matrix Differential Systems with Damping. *Applied Mathematics and Computation*, 170:545-555, 2005.

[22] J. C. Zúñiga. Algorithmes numériques pour les matrices polynomiales avec applications en commande. *Thèse à l'Institut National des Sciences Appliquées de Toulouse*, 2005.