An Algorithm for Computing Jordan Chains and Inverting Analytic Matrix Functions

Jon Wilkening *

May 6, 2007

Abstract

We present an efficient algorithm for obtaining a canonical system of Jordan chains for an $n \times n$ regular analytic matrix function $A(\lambda)$ that is singular at the origin. For any analytic vector function $b(\lambda)$, we show that each term in the Laurent expansion of $A(\lambda)^{-1}b(\lambda)$ may be obtained from the previous terms by solving an $(n+d)\times(n+d)$ linear system, where d is the order of the zero of $\det A(\lambda)$ at $\lambda = 0$. The matrix representing this linear system contains $A(0)$ as a principal submatrix, which can be useful if $A(0)$ is sparse. The last several iterations can be eliminated if left Jordan chains are computed in addition to right Jordan chains. The performance of the algorithm in floating point and exact (rational) arithmetic is reported for several test cases. The method is shown to be forward stable in floating point arithmetic.

Key words. Matrix valued function, Jordan chain, Keldysh chain, analytic perturbation, matrix inversion, Laurent series

AMS subject classifications. 47A56, 47A46, 15A09, 41A58

1 Introduction

Analytic matrix functions arise frequently in mathematics. The most common application [7, 10] involves solving the ordinary differential equation

$$A_l \frac{d^l x}{dt^l} + \cdots + A_0 x = 0 \tag{1.1}$$

using the ansatz $x(t) = [\frac{t^k}{k!}x_0 + \cdots + tx_{k-1} + x_k]e^{\lambda_0 t}$, $(x_0 \neq 0)$, which is a solution if and only if

$$\sum_{m=0}^{k} \frac{t^{k-m}e^{\lambda_0 t}}{(k-m)!} \left[\sum_{j=0}^{m} \frac{1}{j!} A^{(j)}(\lambda_0) x_{m-j}\right] \equiv 0, \tag{1.2}$$

*University of California, Berkeley, CA 94720 (wilken@math.berkeley.edu). The research of the author was supported in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under Contract Number DE-AC03-76SF00098, and by the National Science Foundation through grant DMS-0101439.

where $A(\lambda) = A_0 + \cdots + \lambda^l A_l$. Vectors $x_0, \ldots, x_k$ satisfying the bracketed expression in (1.2) for $m = 0, \ldots, k$ are said to form a (right) Jordan chain of length $k+1$ for $A(\lambda)$ at λ_0. This condition is equivalent to requiring that the vector polynomial

$$x(\lambda) = x_0 + (\lambda - \lambda_0)x_1 + \cdots + (\lambda - \lambda_0)^k x_k \tag{1.3}$$

is a root function of $A(\lambda)$ at λ_0 of order $k+1$, i.e. $A(\lambda)x(\lambda) = O(\lambda - \lambda_0)^{k+1}$.

Analytic matrices also arise in contour integral representations of solutions of various boundary value problems. For example, a solution of the Poisson equation $\Delta u = 1$ in the first quadrant with Dirichlet boundary conditions is given by the formula $u(r,\theta) = r^2/4 + u_S(r,\theta)$, where

$$u_S(r,\theta) = \int_\gamma r^\lambda F(\lambda,\theta) A(\lambda)^{-1} b(\lambda)\, d\lambda, \tag{1.4}$$

γ is a contour surrounding $\lambda_0 = 2$ in the complex plane, $b(\lambda) = \dfrac{1}{\lambda - 2}\begin{pmatrix} -1/4 \\ -1/4 \end{pmatrix}$, $F(\lambda,\theta) = \left(\cos\lambda\theta, \frac{\sin\lambda\theta}{\lambda}\right)$, and

$$A(\lambda) = \begin{pmatrix} 1 & 0 \\ \cos\frac{\lambda\pi}{2} & \frac{1}{\lambda}\sin\frac{\lambda\pi}{2} \end{pmatrix}, \tag{1.5}$$

Note that at $\lambda = 2$, $A(\lambda)$ is singular and $b(\lambda)$ has a pole. The function $u_S(\lambda,\theta)$ may be computed by expanding $A(\lambda)^{-1}b(\lambda)$ in a Laurent expansion and applying the residue theorem, which yields

$$u_S(r,\theta) = r^2\left[\left(\tfrac{\theta}{\pi} - \tfrac{1}{4}\right)\cos 2\theta + \tfrac{1}{\pi}\ln r \sin 2\theta\right]. \tag{1.6}$$

Jordan chains for $A(\lambda)$ are important in this problem because the root functions defined in (1.3) appear as poles of $A(\lambda)^{-1}b(\lambda)$, and may be used to recursively compute an arbitrary number of terms in the Laurent expansion; see Section 4. Singularities of general elliptic systems in the plane near corners and interface junctions have representations analogous to (1.4); see [4].

Other applications of analytic matrix functions include stochastic matrices and Markov chains [11], linear vibrations [13], singular differential equations [3], dynamical systems [16], and control theory [3, 7].

The local behavior of a matrix or operator $A(\lambda)$ that depends analytically on λ has been studied extensively [2, 9, 12], especially in the case that $A(\lambda)$ is a matrix pencil $A_0 + \lambda A_1$ [7, 8, 14, 17] or a matrix polynomial $A_0 + \cdots + \lambda^l A_l$ [10, 15]. Monic ($A_l = I$) matrix polynomials are often reduced to the matrix pencil case via a standard linearization procedure [7, 10]. However, for analytic matrices such as (1.5) above, the expansion of $A(\lambda)$ does not terminate, and any truncation may have a singular term A_l of highest order; hence, an approach based on linearization seems inappropriate in this context. Truncating the expansion and computing the Smith form [10] is also an option, but this is very expensive; see Section 4.2.

In [1], Avrachenkov et. al. present an algorithm for expanding the inverse of an analytic matrix function in a Laurent series. More specifically, they show how to reduce a system of matrix equations (involving $n \times n$ matrices) of the form

$$\begin{aligned} A_0 X_0 &= B_0 \\ &\cdots \\ A_0 X_{s-1} + \cdots + A_{s-1} X_0 &= B_{s-1} \end{aligned} \qquad \begin{pmatrix} A_k, B_k \text{ given} \\ X_k \text{ unknown} \end{pmatrix} \tag{1.7}$$

to another system of this form involving $p \times p$ matrices, where $p = \dim \ker A_0$ is the geometric multiplicity of the origin. Let d be its algebraic multiplicity. Our approach employs a different type of reduction based on adding d rows and columns to A_0 to construct an invertible matrix that can be used to recursively solve for successive terms in the Laurent expansion of $A(\lambda)^{-1}$. It also provides complete Jordan chain information, which is often important in applications.

Canonical systems of Jordan chains depend discontinuously on the leading coefficients $A_0, \ldots, A_{s-1}$ of $A(\lambda)$, where $A_k = \frac{1}{k!} A^{(k)}(0)$ and s is the order of the pole of $A(\lambda)^{-1}$ at $\lambda = 0$. Nevertheless, there are many applications (e.g. in algebraic geometry or partial differential equations) where these coefficients are known *exactly*. In such cases our algorithm can be run using rational arithmetic to solve the problem analytically. In finite precision arithmetic, the method is *forward stable* when nullspaces are computed using a threshold on the smallest singular values. The condition number depends on how close $A(\lambda)$ is to an analytic matrix function that is even more singular at the origin; see Appendix A.

Numerical stability is possible because we have specified the location of the eigenvalue of interest (e.g. the origin) as part of the problem statement. The eigenvalue itself cannot be computed in floating point arithmetic (unless it is simple) as roundoff error is likely to split the eigenvalue like $O(\varepsilon^{1/d})$, where ε is machine precision and d is the algebraic multiplicity of the eigenvalue. Even using extra precision, it is impossible to determine affirmatively that a tight cluster of eigenvalues should exactly coalesce unless their multiplicities (or locations) are determined algebraically. The stability of this algorithm is useful in applications with a simple, parasitic eigenvalue λ_1 close to a known eigenvalue λ_0. For example, integer values of λ_0 occur frequently in the corner singularities problem described above. By computing a canonical system of Jordan chains at λ_0, we learn the algebraic multiplicity of λ_0 and can use this information to compute λ_1 by a contour integral technique without corruption from λ_0.

2 Canonical systems of root functions

In this section we establish notation and summarize the local spectral theory of analytic matrix functions; the presentation roughly follows [9].

Suppose $A(\lambda)$ is an $n \times n$ matrix whose entries depend analytically on λ in a neighborhood of the origin. Suppose $\Delta(\lambda) := \det A(\lambda)$ has a zero of order $d \geq 1$ at the origin. In particular, we assume that $A(\lambda)$ is regular, i.e. $\Delta(\lambda)$ is not identically

zero. A vector polynomial

$$x(\lambda) = x_0 + \lambda x_1 + \cdots + \lambda^{\kappa-1} x_{\kappa-1}, \qquad (x_k \in \mathbb{C}^n,\ \kappa \geq 1) \tag{2.1}$$

is called a root function of $A(\lambda)$ of order κ at $\lambda = 0$ if

$$A(\lambda)x(\lambda) = O(\lambda^\kappa) \quad \text{and} \quad x_0 \neq 0. \tag{2.2}$$

This is equivalent to requiring that $(x_0; \ldots; x_{\kappa-1}) \in \ker \mathbb{A}_{\kappa-1}$, where $(x; y) = (x^T, y^T)^T$ and the matrices $\mathbb{A}_k$ are defined via

$$\mathbb{A}_k = \begin{pmatrix} A_0 & 0 & \cdots & 0 \\ A_1 & A_0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ A_k & A_{k-1} & \cdots & A_0 \end{pmatrix}, \qquad A_j = \frac{1}{j!} A^{(j)}(0). \tag{2.3}$$

Here we think of $\mathbb{C}^{n\kappa}$ as representing the space $\mathcal{P}/\lambda^\kappa\mathcal{P}$, where $\mathcal{P}$ is the space of formal power series in λ with coefficients in $\mathbb{C}^n$ and we identify two power series with the same leading terms $x_0 + \lambda x_1 + \cdots + \lambda^{\kappa-1} x_{\kappa-1}$. In this representation, the action of $A(\lambda)$ on such a power series becomes a matrix-vector multiplication $\mathbb{A}_{\kappa-1} \cdot (x_0; \ldots; x_{\kappa-1})$.

The vectors x_k in (2.1) are said to form a Jordan chain (or Keldysh chain) of length κ. Several vector polynomials $x_j(\lambda)$ form a system of root functions if

$$x_j(\lambda) = x_{j,0} + \lambda x_{j,1} + \cdots + \lambda^{\kappa_j - 1} x_{j,\kappa_j - 1}, \qquad (1 \leq j \leq p) \tag{2.4}$$

$$A(\lambda)x_j(\lambda) = O(\lambda^{\kappa_j}), \tag{2.5}$$

$$\text{the set } \{x_{j,0}\}_{j=1}^p \text{ is linearly independent in } \mathbb{C}^n. \tag{2.6}$$

Such a system is said to be *canonical* if $p = \dim\ker A_0$, $x_1(\lambda)$ is a root function of maximal order κ_1, and for $j > 1$, $x_j(\lambda)$ has maximal order κ_j among all root functions $x(\lambda)$ such that $x(0)$ is linearly independent of $x_1(0), \ldots, x_{j-1}(0)$ in $\mathbb{C}^n$. The resulting sequence of numbers $\kappa_1 \geq \cdots \geq \kappa_p$ is uniquely determined by $A(\lambda)$; see [9] and Section 3.

As a simple example, suppose $A(\lambda) = A_0 - \lambda I$. Then x_0, …, $x_{\kappa-1}$ form a Jordan chain of length κ at $\lambda = 0$ iff $x_0 \neq 0$ and

$$A_0 x_0 = 0, \qquad A_0 x_k = x_{k-1} \qquad (1 \leq k \leq \kappa - 1) \tag{2.7}$$

In this case x_0, …, $x_{\kappa-1}$ are linearly independent and can be extended to a basis matrix $U = [x_0, \ldots, x_{n-1}]$ that puts A in Jordan canonical form: $U^{-1}AU = J$. (This requires knowing all the eigenvalues of A and finding a canonical system of Jordan chains at each). But in general the x_k need not be linearly independent (some can even be zero) and it is more useful to reduce A to local Smith form $A(\lambda) = C(\lambda)D(\lambda)E(\lambda)^{-1}$ as we now explain.

For any system of root functions $x_j(\lambda)$ (not necessarily canonical), we may choose constant vectors $x_{p+1}, \ldots, x_n$ that extend $x_1(0), \ldots, x_p(0)$ to a basis for

$\mathbb{C}^n$. We call the system $x_1(\lambda), \ldots, x_p(\lambda), x_{p+1}, \ldots, x_n$ an extended system of root functions, and set $\kappa_{p+1} = \cdots = \kappa_n = 0$. Next we define the matrices

$$E(\lambda) = \big[x_1(\lambda), \ldots, x_n(\lambda)\big], \tag{2.8}$$

$$D(\lambda) = \operatorname{diag}(\lambda^{\kappa_1}, \ldots, \lambda^{\kappa_n}), \tag{2.9}$$

$$C(\lambda) = A(\lambda)E(\lambda)D(\lambda)^{-1}. \tag{2.10}$$

By (2.5), all singularities of $C(\lambda)$ at the origin are removable. Since $E(0)$ is invertible, $\det C(\lambda)$ has a zero of order $d - \sum_{j=1}^n \kappa_j$ at the origin, where d is the order of the zero of $\Delta(\lambda)$ at the origin. The following theorem is proved in [9]:

THEOREM 2.1 *The following three conditions are equivalent:*

(1) the columns $x_j(\lambda)$ of $E(\lambda)$ form an extended canonical system of root functions for $A(\lambda)$ at $\lambda = 0$ with orders $\kappa_1 \geq \cdots \geq \kappa_n \geq 0$.

(2) $C(0)$ is invertible.

(3) $\sum_{j=1}^n \kappa_j = d$, where d is the algebraic multiplicity of $\lambda = 0$

Canonical systems of root functions are useful due to the fact that they completely characterize the meromorphic behavior of $A(\lambda)^{-1}$. Observe that

$$A(\lambda)^{-1} = E(\lambda)D(\lambda)^{-1}C(\lambda)^{-1}, \tag{2.11}$$

where $E(\lambda)$ and $C(\lambda)$ are invertible at $\lambda = 0$. Thus for any holomorphic vector function $b(\lambda)$, all poles of $A(\lambda)^{-1}b(\lambda)$ at the origin are linear combinations of the functions $\lambda^{-1-k}x_j(\lambda)$ with $1 \leq j \leq p$ and $0 \leq k \leq \kappa_j - 1$. Moreover, each of these functions actually occurs as a pole. For example, if we define $b_{jk}(\lambda)$ as the unique vector polynomial of degree $\kappa_1 - 1$ such that

$$b_{jk}(\lambda) = \lambda^{-1-k}A(\lambda)[x_{j,0} + \cdots + \lambda^k x_{j,k}] + O(\lambda^{\kappa_1}), \tag{2.12}$$

then

$$A(\lambda)^{-1}b_{jk}(\lambda) = \lambda^{-1-k}x_j(\lambda) + O(1), \qquad (1 \leq j \leq p, \quad 0 \leq k \leq \kappa_j - 1). \tag{2.13}$$

Other shifted and truncated versions of the residual $A(\lambda)x_j(\lambda)$ can also be used for $b_{jk}(\lambda)$, but the choice in (2.12) is relevant to Algorithm 4.6 below.

3 An algorithm for computing Jordan chains

In the previous section, we described the standard theoretical (as opposed to algorithmic) construction of canonical systems of root functions for $A(\lambda)$ at $\lambda = 0$. This approach can be thought of as a depth-first approach: $x_1(\lambda)$ is chosen to be any root function of maximal order κ_1, and the remaining root functions $x_j(\lambda)$ are chosen to be of maximal order κ_j such that $x_j(0)$ is linearly independent of the previously accepted vectors $x_0(0)$, ..., $x_{j-1}(0)$.

Our algorithm uses a breadth-first approach. Instead of starting with a single root function of maximal order, we start with the space of all root functions of order at least 1 (namely $\ker A_0$). We then use this space to find all root functions of order at least 2, then 3, etc., until there are no root functions of higher order. At the kth stage of this process (starting at $k = 0$), we extract the Jordan chains of length exactly $k + 1$, thereby obtaining a canonical system $x_j(\lambda)$ in reverse j order (from p to 1).

In more detail, let us define the subspaces $\mathbb{W}_k$ and the canonical injections $\iota : \mathbb{C}^{nk} \to \mathbb{C}^{n(k+1)}$ via

$$\mathbb{W}_k = \ker \mathbb{A}_k \subset \mathbb{C}^{n(k+1)}, \qquad \iota(x_0; \dots; x_{k-1}) = (0; x_0; \dots; x_{k-1}). \tag{3.1}$$

The block structure of the augmented matrices $\mathbb{A}_k$ defined in (2.3) implies that $\iota(\mathbb{W}_{k-1}) \subset \mathbb{W}_k$ for $k \geq 1$. A Jordan chain of length $k + 1$ may be thought of as a non-zero element of the space

$$W_k = \mathbb{W}_k / \iota(\mathbb{W}_{k-1}), \tag{3.2}$$

where we take $\iota(\mathbb{W}_{-1})$ to mean $\{0\} \subset \mathbb{C}^n$. Evidently, the projection $\pi_0 : \mathbb{C}^{n(k+1)} \to \mathbb{C}^n$ given by

$$\pi_0(x_0; \dots; x_k) = x_0 \tag{3.3}$$

is injective from W_k to $\mathbb{C}^n$, so several vectors in W_k are linearly independent iff their zeroth terms are linearly independent in $\mathbb{C}^n$. To count Jordan chains, it is useful to define

$$R_k = \dim(\mathbb{W}_k), \qquad r_k = \dim(W_k), \qquad k \geq 0 \tag{3.4}$$

so that $R_k = r_k + \cdots + r_0$ and $r_k \leq \cdots \leq r_0$. These last inequalities follow from the fact that the truncation operator $\tau : \mathbb{C}^{n(k+2)} \to \mathbb{C}^{n(k+1)}$ given by

$$\tau(x_0; \dots; x_{k+1}) = (x_0; \dots; x_k) \tag{3.5}$$

injectively maps W_{k+1} into W_k, i.e. shortening a Jordan chain gives another valid Jordan chain. Finally, to extract all Jordan chains of length $k + 1$, we choose vectors

$$\tilde{x}_{j,\cdot} = (\tilde{x}_{j,0}; \dots; \tilde{x}_{j,k}) \in \mathbb{W}_k, \qquad (j = r_{k+1} + 1, \dots, r_k) \tag{3.6}$$

such that the equivalence classes $[\tilde{x}_{j,\cdot}] \in W_k$ are linearly independent and complement the subspace $\tau(W_{k+1}) \subset W_k$ consisting of truncations of longer chains. We also set $\kappa_j = k + 1$ for each j in (3.6). When $r_{k+1} = r_k$, the range of indices in (3.6) is empty and there are no partial multiplicities κ_j equal to $k + 1$. The algorithm terminates as soon as $r_k = 0$, which happens when $k = s := \kappa_1$, the maximal Jordan chain length. Since $r_k - r_{k+1} = \#\{j : \kappa_j = k + 1\}$ and $r_s = 0$, we find that

$$\begin{aligned} \kappa_1 + \cdots + \kappa_p &= s(r_{s-1} - r_s) + (s-1)(r_{s-2} - r_{s-1}) + \cdots + 1(r_0 - r_1) \\ &= r_{s-1} + \cdots + r_0 = R_{s-1} = d, \end{aligned} \tag{3.7}$$

ALGORITHM 3.1 (Jordan chains)

$k = 0$
$X_0 = \mathbb{X}_0 = \ker A_0$ (i.e. columns form a basis for the kernel)
while $r_k > 0$ (r_k, R_k = # of columns in $X_k, \mathbb{X}_k$)
  $[V_k; U_k] = \ker \mathcal{A}_{k+1}$ (U_k is $R_k \times r_{k+1}$, V_k is $n \times r_{k+1}$)
  $X_{k+1} = [\mathbb{X}_k U_k; V_k]$
  $\mathbb{X}_{k+1} = [\iota(\mathbb{X}_k), X_{k+1}]$ (ι acts on matrices column by column)
  U'_k = last r_k rows of U_k (U'_k is $r_k \times r_{k+1}$)
  $\widetilde{U}_k$ = any extension of the columns of U'_k to a basis for $\mathbb{C}^{r_k}$
  $\widetilde{X}_k = X_k \widetilde{U}_k$ (cols of $\widetilde{X}_k$ are J-chains of length $k+1$)
  $k = k + 1$
$s = k$ ($s = \kappa_1$ = maximal Jordan chain length)
return $\widetilde{X}_{s-1}, \ldots, \widetilde{X}_0$

where $p = r_0$ is the geometric multiplicity of the origin, d is the algebraic multiplicity, and the final equality follows from Theorem 2.1.

In Algorithm 3.1, we represent $\mathbb{W}_k = \ker \mathbb{A}_k$ and $W_k = \mathbb{W}_k / \iota(\mathbb{W}_{k-1})$ using matrices whose columns span these spaces:

$$\begin{aligned} \operatorname{colspan} \mathbb{X}_k = \mathbb{W}_k, \qquad \operatorname{colspan} X_k &= \{(x_0; x) \in \mathbb{W}_k \ : \ x \perp \mathbb{W}_{k-1}\} \\ &= \ker([\mathbb{A}_k;\ \iota(\mathbb{X}_{k-1})^*]). \end{aligned} \tag{3.8}$$

Here $\iota(\mathbb{X}_{k-1})^*$ is the conjugate transpose of the matrix $\iota(\mathbb{X}_{k-1})$. Structurally, we have

$$\mathbb{X}_k = [\iota(\mathbb{X}_{k-1}), X_k] = \left[\begin{pmatrix} 0_{nk \times r_0} \\ X_0 \end{pmatrix}, \begin{pmatrix} 0_{n(k-1) \times r_1} \\ X_1 \end{pmatrix}, \ldots, \begin{pmatrix} X_k \end{pmatrix} \right]. \tag{3.9}$$

Rather than compute $\ker([\mathbb{A}_{k+1}; \iota(\mathbb{X}_k)^*])$ directly to obtain X_{k+1}, we make use of the fact that shortening a Jordan chain gives another valid Jordan chain in the equivalence class (but not necessarily satisfying the orthogonality condition); hence, there are matrices U_k and V_k such that $X_{k+1} = (\mathbb{X}_k U_k; V_k)$. In this representation, the condition $X_{k+1} = \ker([\mathbb{A}_{k+1}; \iota(\mathbb{X}_k)^*])$ is equivalent to

$$\overbrace{\begin{pmatrix} A_{k+1} & A_k & \cdots & A_0 \\ 0 & (& \mathbb{X}_k^* &) \end{pmatrix}}^{\mathcal{A}_{k+1}} \underbrace{\begin{pmatrix} 0 & \mathbb{X}_k \\ I_{n \times n} & 0 \end{pmatrix} \begin{pmatrix} V_k \\ U_k \end{pmatrix}}_{X_{k+1}} = \begin{pmatrix} 0_{n \times r_{k+1}} \\ 0_{R_k \times r_{k+1}} \end{pmatrix}. \tag{3.10}$$

Thus, we need only find the kernel of the $(n + R_k) \times (n + R_k)$ matrix $\mathcal{A}_{k+1}$ to determine U_k, V_k and r_{k+1}, which immediately give X_{k+1}. Note that $\mathcal{A}_0 = A_0$, $\mathcal{A}_s$

is invertible (with $s = \kappa_1$), and these matrices are nested in the sense that $\mathcal{A}_k$ is a principal submatrix of $\mathcal{A}_{k+1}$ for $k \geq 0$:

$$\mathcal{A}_{k+1} = \begin{array}{c|cc} & n & R_k \\ \hline n & A_0 & [A_{k+1},\dots,A_1]\mathbb{X}_k \\ R_k & \pi_k(\mathbb{X}_k)^* & \iota(\tau(\mathbb{X}_k))^*\mathbb{X}_k \end{array} = \begin{array}{c|ccc} & n+R_{k-1} & & r_k \\ \hline n+R_{k-1} & \multicolumn{2}{c|}{\mathcal{A}_k} & [A_{k+1},\dots,A_1]X_k & n \\ r_k & \pi_k(X_k)^* & \tau(X_k)^*\mathbb{X}_{k-1} & \iota(\tau(\mathbb{X}_k))^*X_k = 0 & R_k \\ & n & R_{k-1} & r_k \end{array} \tag{3.11}$$

Here $\pi_k(x_0;\dots;x_k) = x_k$ and R_{-1} is taken to be zero. This nested structure reduces the work required to form these matrices, and allows a substantial part of the computation of $\ker(\mathcal{A}_k)$ to be re-used when computing $\ker(\mathcal{A}_{k+1})$. Moreover, if A_0 is sparse and $d = R_{s-1}$ is small in comparison to n, each $\mathcal{A}_k$ will also be sparse. Kernels may easily be found symbolically by reducing to row-echelon form, or in floating point arithmetic using the singular value decomposition. Re-use may be achieved in the former case using a modified pivoting strategy, and in the latter case using Givens rotations to quickly restore bi-diagonal form, which is the first stage of the SVD algorithm [7]. If the algorithm used to compute kernels returns orthonormal vectors (i.e. in the floating point case), then the matrices X_k and $\mathbb{X}_k$ will have orthonormal columns for all k.

It remains to explain the final step in which Jordan chains of length $k+1$ are extracted. As discussed after (3.6) above, this is equivalent to extending the columns of $\tau(X_{k+1})$ to a basis for W_k. Since the columns of X_k are already a basis for W_k, the two bases must be related to each other by an invertible transformation. But linear independence of cosets in W_k is equivalent to linear independence of their zeroth terms in $\mathbb{C}^n$. It follows that the columns of $\widetilde{X}_k$ provide such an extension iff they belong to $\mathbb{W}_k$ and there is an invertible matrix E_k such that

$$\pi_0([\tau(X_{k+1}), \widetilde{X}_k]) = \pi_0(X_k)E_k, \qquad (k \geq 0). \tag{3.12}$$

The formulas $X_{k+1} = [\mathbb{X}_k U_k; V_k]$ and $\mathbb{X}_k = [\iota(\mathbb{X}_{k-1}), X_k]$ ensure that

$$\pi_0(\tau(X_{k+1})) = \pi_0(X_{k+1}) = \pi_0(\mathbb{X}_k)U_k = \pi_0(X_k)U'_k, \tag{3.13}$$

where U'_k consists of the last r_k rows of the $R_k \times r_{k+1}$ matrix U_k. As the columns of $\pi_0(X_{k+1})$ are linearly independent, the columns of U'_k must also be linearly independent, and there is an $r_k \times (r_k - r_{k+1})$ matrix $\widetilde{U}_k$ such that $E_k = [U'_k, \widetilde{U}_k]$ is invertible. The matrix $\widetilde{X}_k = X_k\widetilde{U}_k$ then satisfies (3.12), as required.

REMARK 3.2 If $r_{k+1} = r_k$, then $\widetilde{U}_k$ and $\widetilde{X}_k$ are empty matrices with zero columns, and there are no partial multiplicities κ_j equal to $k+1$.

REMARK 3.3 The process of extending U'_k to an invertible matrix $[U'_k, \widetilde{U}_k]$ is easily done in many ways, e.g. using the QR factorization or SVD of U'_k, or by row reducing the adjoint matrix $(U'_k)^*$ symbolically to find its kernel.

EXAMPLE 3.4 Let $A(\lambda) = \begin{pmatrix} \lambda & -\lambda^2 \\ \lambda^2 & 0 \end{pmatrix} = A_0 + \lambda A_1 + \lambda^2 A_2$. In this case, it is simplest to compute $X_k = \ker([\mathbb{A}_k; \iota(\mathbb{X}_{k-1})^*])$ directly, which yields

$$X_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X_1 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad X_2 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad X_3 = \varnothing_{8\times 0}. \tag{3.14}$$

U_k and V_k may be read off from $X_{k+1} = (\mathbb{X}_k U_k; V_k)$: $U_0 = (0;1)$, $V_0 = (0;0)$, $U_1 = (1;0;1)$, $V_1 = (0;0)$, $U_2 = \varnothing_{4\times 0}$, $V_2 = \varnothing_{2\times 0}$. We then obtain $\widetilde{X}_0 = (1;0)$, $\widetilde{X}_1 = \varnothing_{4\times 0}$, $\widetilde{X}_2 = X_2$, so a canonical system of root functions is

$$x_1(\lambda) = \begin{pmatrix} \lambda \\ 1 \end{pmatrix}, \quad \kappa_1 = 3, \qquad x_2(\lambda) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \kappa_2 = 1. \tag{3.15}$$

This example shows (with $k = 1$) that although the columns of $\tau(X_{k+1})$ are guaranteed to be linearly independent modulo $\iota(\mathbb{W}_{k-1})$, they need not satisfy the orthogonality condition required to belong to $\operatorname{colspan}(X_k)$.

EXAMPLE 3.5 Let $A(\lambda) = \begin{pmatrix} 2\lambda & -\lambda & i\lambda \\ -2i\lambda & 1 & 2\lambda \\ -2i & i-\lambda & 1 \end{pmatrix} = A_0 + \lambda A_1$. We find that $r_0 = 1$, $r_1 = 1$, $r_2 = 1$ and $r_3 = 0$ so that $s = 3$. We also obtain

$$\mathcal{A}_3 = \left(\begin{array}{ccc|c|cc} 0 & 0 & 0 & 0 & 0 & 98i \\ 0 & 1 & 0 & -2 & -12i & 180 \\ -2i & i & 1 & 0 & -10 & -52i \\ \hline -i & 0 & -2 & 0 & 0 & 0 \\ \hline 4 & 10 & 2i & 25 & 0 & 0 \\ \hline -60i & -52i & 30 & 164i & 2401 & 0 \end{array}\right), \quad \mathbb{X}_2 = \begin{pmatrix} 0 & 0 & 49i \\ 0 & 0 & 0 \\ 0 & 0 & -98 \\ 0 & 5i & 72 \\ 0 & 0 & 98 \\ 0 & -10 & 46i \\ i & 4 & 60i \\ 0 & 10 & 52i \\ -2 & -2i & 30 \end{pmatrix}, \tag{3.16}$$

$U_0 = (5)$, $V_0 = (4;10;-2i)$, $U_1 = \left(-\frac{164}{5}i; \frac{49}{5}\right)$, $V_1 = (60i; 52i; 30)$, $U_2 = \varnothing_{3\times 0}$, $V_2 = \varnothing_{3\times 0}$, $X_3 = \varnothing_{12\times 0}$. Note that $(V_k; U_k)$ are the null vectors for the matrices $\mathcal{A}_{k+1}$. Since $\widetilde{X}_2 = X_2$ is the only non-empty Jordan chain matrix,

$$x_1(\lambda) = \begin{pmatrix} 49i + 72\lambda + 60i\lambda^2 \\ 98\lambda + 52i\lambda^2 \\ -98 + 46i\lambda + 30\lambda^2 \end{pmatrix}, \qquad \kappa_1 = 3 \tag{3.17}$$

is a canonical system of root functions. A simpler result can be obtained using $X_1 = (i;0;-2;1;2;0)$ and $X_2 = (i;0;-2;1;2;0;2i;2i;0)$, but these do not satisfy the orthogonality condition $\iota(\mathbb{X}_{k-1})^* X_k = 0$. Nevertheless, the algorithm remains correct if arbitrary vectors in $\iota(\mathbb{W}_{k-1})$ are added to the columns of X_k. This can

be useful when the algorithm is run in exact arithmetic to reduce the size of the rational entries in $\mathcal{A}_k$ and $\mathbb{X}_k$ by setting certain components of X_k to zero rather than requiring orthogonality; see Section 4.1. Different choices of the X_k lead to different matrices $\mathcal{A}_k$ and to different root functions $x_j(\lambda)$, but to the same terms in the Laurent expansion of $A(\lambda)^{-1}$ in Section 4. In floating point arithmetic, requiring orthogonality seems most natural (and stable: see Appendix A).

4 Laurent expansion of $A(\lambda)^{-1}$

As discussed in Section 2, the meromorphic behavior of $A(\lambda)^{-1}$ is completely characterized by any canonical system of Jordan chains. In this section, we show how to use the output of Algorithm 3.1 to efficiently compute an arbitrary number of terms in the Laurent expansion of $A(\lambda)^{-1}b(\lambda)$ for any analytic (or meromorphic) vector function $b(\lambda)$.

Let $s = \kappa_1$ denote the maximal Jordan chain length. This is the first index for which X_s has no columns, $\mathbb{X}_s = \iota(\mathbb{X}_{s-1})$, and the linear system

$$\mathcal{A}_s \begin{pmatrix} v \\ u \end{pmatrix} = \begin{pmatrix} A_s & A_{s-1} & \cdots & A_0 \\ 0_{d\times n} & (& \mathbb{X}_{s-1}^* &) \end{pmatrix} \underbrace{\begin{pmatrix} 0_{sn\times n} & \mathbb{X}_{s-1} \\ I_{n\times n} & 0_{n\times d} \end{pmatrix} \begin{pmatrix} v \\ u \end{pmatrix}}_{(x_0;\ldots;x_s)} = \begin{pmatrix} b \\ 0 \end{pmatrix} \tag{4.1}$$

is uniquely solvable and provides a linear mapping $b \mapsto (v;u) \mapsto (x_0;\ldots;x_s)$. Here $d = R_{s-1}$ is the algebraic multiplicity of the origin, $x_k, b, v \in \mathbb{C}^n$ and $0, u \in \mathbb{C}^d$. We will denote this mapping by

$$(x_0;\ldots;x_s) = \operatorname{solve}(b) := \begin{pmatrix} 0 & \mathbb{X}_{s-1} \\ I & 0 \end{pmatrix} \mathcal{A}_s^{-1} \begin{pmatrix} b \\ 0 \end{pmatrix}. \tag{4.2}$$

Suppose we wish to compute the first $q+1$ terms in the Laurent expansion of $A(\lambda)^{-1}b(\lambda)$, where $b(\lambda) = b_0 + \lambda b_1 + \lambda^2 b_2 + \cdots$. Equation (2.11) shows that the most singular term in the expansion of $A(\lambda)^{-1}$ is λ^{-s}. If we can find any $x(\lambda) = x_0 + \lambda x_1 + \cdots + \lambda^{q+s} x_{q+s}$ such that

$$A(\lambda)[\lambda^{-s}x_0 + \cdots + \lambda^q x_{q+s}] = b_0 + \cdots + \lambda^q b_q + O(\lambda^{q+1}), \tag{4.3}$$

then

$$A(\lambda)^{-1}b(\lambda) = \lambda^{-s}[x(\lambda) + O(\lambda^{q+1})], \tag{4.4}$$

so the terms $x_0, \ldots, x_q$ are correctly determined by solving (4.3). Matching like powers of λ in (4.3), the x_k should satisfy

$$\mathbb{A}_{q+s}(x_0;\ldots;x_{q+s}) = (0;\ldots;0;b_0;\ldots;b_q). \tag{4.5}$$

To verify that this equation has a solution, observe that for $q \geq 0$ we have $\dim\ker \mathbb{A}_{q+s}^* = \dim\ker \mathbb{A}_{s-1}^*$; hence, the block structure of $\mathbb{A}_{q+s}^*$ ensures that

$$(x_0;\ldots;x_{q+s}) \in \ker \mathbb{A}_{q+s}^* \quad \Rightarrow \quad x_s = \cdots = x_{q+s} = 0, \qquad (q \geq 0,\ x_k \in \mathbb{C}^n). \tag{4.6}$$

As a result, the right-hand side of (4.5) belongs to $\ker(\mathbb{A}_{q+s}^*)^\perp$, as required. Note that any two solutions of (4.5) differ by at most a vector $(0;\ldots;0;\xi_0;\ldots;\xi_{s-1}) \in \mathbb{C}^{(q+1+s)n}$ with $(\xi_0;\ldots;\xi_{s-1}) \in \operatorname{colspan}(\mathbb{X}_{s-1})$. This shows to what extent the terms $x_{q+1},\ldots,x_{q+s}$ are left undetermined in (4.3).

Next we show how to bootstrap a solution to (4.5) by repeatedly solving the linear system (4.1). Suppose $k \geq 1$ and we have found vectors x_i satisfying

$$\mathbb{A}_{k+s-1}(x_0;\ldots;x_{k+s-1}) = (0;\ldots;0;b_0;\ldots;b_{k-1}). \tag{4.7}$$

For example, if $k = 1$ then such a solution is given by $(x_0;\ldots;x_s) = \operatorname{solve}(b_0)$. We wish to find corrections ξ_i such that

$$\begin{pmatrix} A_0 & 0 & \cdots & 0 \\ A_1 & A_0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ A_{k+s} & A_{k+s-1} & \cdots & A_0 \end{pmatrix} \begin{pmatrix} x_0 \\ \cdots \\ x_{k-1} \\ x_k + \xi_0 \\ \cdots \\ x_{k+s-1} + \xi_{s-1} \\ \xi_s \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ b_0 \\ \vdots \\ b_k \end{pmatrix}. \tag{4.8}$$

Moving the known vectors x_i to the right hand side, this is equivalent to solving

$$\mathbb{A}_{k+s}(0;\ldots;0;\xi_0;\ldots;\xi_s) = (0;\ldots;0;b_k - A_{k+s}x_0 - \cdots - A_1 x_{k+s-1}). \tag{4.9}$$

A solution of this equation is given by $(\xi_0;\ldots;\xi_s) = \operatorname{solve}\left(b_k - \sum_{j=1}^{k+s} A_j x_{k+s-j}\right)$, which we use to correct the x_i in (4.8):

ALGORITHM 4.1 (Laurent expansion of $A(\lambda)^{-1}b(\lambda)$, first $q+1$ terms)

determine s, $\mathcal{A}_s$, $\mathbb{X}_{s-1}$ using Algorithm 3.1
$x_0 = \cdots = x_{q+s} = 0$
for $k = 0,\ldots,q$
  $(\xi_0;\ldots;\xi_s) = \operatorname{solve}\left[b_k - \sum_{j=1}^{k+s} A_j x_{k+s-j}\right]$, (omit sum if $k = 0$)
  for $i = 0,\ldots,s$
    $x_{k+i} = x_{k+i} + \xi_i$
return $(x_0;\ldots;x_q)$

REMARK 4.2 If A_0 is invertible (so that $s = d = 0$), this algorithm reduces to the well-known recursion $x_k = A_0^{-1}\left[b_k - \sum_{j=1}^{k} A_j x_{k-j}\right]$, $k = 0,\ldots,q$.

REMARK 4.3 After the final iteration, the minimum norm solution of (4.5) has been found:

$$(x_0,\ldots,x_{q+s}) = \operatorname{pinv}(\mathbb{A}_{q+s})(0;\ldots;0;b_0;\ldots;b_q). \tag{4.10}$$

To see this, note that each time through the loop, $(\xi_1;\dots;\xi_s) \perp \mathbb{W}_{s-1}$ due to the last d equations of (4.1). By (3.9), we conclude that

$$\xi_s \perp \mathbb{W}_0 \quad , \quad (\xi_{s-1};\xi_s) \perp \mathbb{W}_1 \quad , \quad \dots \quad , \quad (\xi_1;\dots;\xi_s) \perp \mathbb{W}_{s-1}. \tag{4.11}$$

But any $(w_0;\dots;w_{s-1}) \in \mathbb{W}_{s-1}$ also satisfies $w_0 \in \mathbb{W}_0$, $(w_0;w_1) \in \mathbb{W}_1$, etc. Thus we have

$$\begin{pmatrix} \xi_s \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} \xi_{s-1} \\ \xi_s \\ 0 \\ \vdots \end{pmatrix}, \dots, \begin{pmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_s \end{pmatrix} \in (\mathbb{W}_{s-1})^\perp. \tag{4.12}$$

Since $(x_{q+1};\dots;x_{q+s})$ is a sum of such vectors, it belongs to $(\mathbb{W}_{s-1})^\perp$, as claimed.

REMARK 4.4 Several solutions may be found simultaneously by adding additional columns to the vectors $b(\lambda)$ and $x(\lambda)$ above. In particular, replacing $b(\lambda)$ by $I_{n\times n}$ and $x(\lambda)$ by $Q(\lambda)$, we obtain $A(\lambda)^{-1} = \lambda^{-s}[Q_0 + \lambda Q_1 + \lambda^2 Q_2 + \cdots]$. However, in practice one often wishes to apply $A(\lambda)^{-1}$ to a single vector function $b(\lambda)$, in which case it is more efficient to carry out the procedure directly rather than compute the Laurent expansion of $A(\lambda)^{-1}$ first. See Section 4.2 below.

EXAMPLE 4.5 Let $A(\lambda)$ be defined as in Example 3.5. Then $s = 3$ and $\mathcal{A}_3$, $\mathbb{X}_2$ were given in (3.16) above. For any $q \geq 2$, Algorithm 4.1 gives

$$Q_0 = \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 \\ i & 0 & 0 \end{pmatrix}, \quad Q_1 = \begin{pmatrix} -i & 0 & -\frac{i}{2} \\ -i & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}, \quad Q_2 = \begin{pmatrix} 1 & -\frac{i}{2} & -1 \\ 0 & 0 & -1 \\ i & 1 & -i \end{pmatrix}, \tag{4.13}$$

and $Q_{3\leq k\leq q} = 0_{3\times 3}$, i.e. the Laurent expansion $A(\lambda)^{-1} = \lambda^{-3}[Q_0 + \lambda Q_1 + \cdots]$ terminates at Q_2.

4.1 Exact arithmetic and left Jordan chains

When Algorithm 4.1 is performed in infinite precision using rational numbers, the cost of each operation also grows with the problem size, so a straightforward $O(n^3)$ operation count will underestimate the running time. We have found that two modifications can significantly reduce the size of the rational numbers that arise in the computation. The first modification involves the last R_k rows of $\mathcal{A}_{k+1}$. We define the first n rows as before, set $v_j = \mathcal{A}_{k+1}(1{:}n, j)$ and define $j_1, \dots, j_{R_k}$ to be the first R_k indices encountered for which v_j is linearly dependent on $v_1, \dots, v_{j-1}$. Then we define row $n+i$ of $\mathcal{A}_{k+1}$ to have a one in column j_i and zeros in all other

columns. In Example 3.5, the resulting matrices $\mathcal{A}_s$ and $\mathbb{X}_{s-1}$ are

$$\mathcal{A}_3 = \left(\begin{array}{ccc|c|cc} 0 & 0 & 0 & 0 & 0 & 2i \\ 0 & 1 & 0 & -2 & -2i & 4 \\ -2i & i & 1 & 0 & -2 & -2i \\ \hline 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right), \quad \mathbb{X}_2 = \begin{pmatrix} 0 & 0 & i \\ 0 & 0 & 0 \\ 0 & 0 & -2 \\ 0 & i & 1 \\ 0 & 0 & 2 \\ 0 & -2 & 0 \\ i & 1 & 2i \\ 0 & 2 & 2i \\ -2 & 0 & 0 \end{pmatrix}. \tag{4.14}$$

This change sacrifices the minimum norm property of the solution of (4.5) found by Algorithm 4.1 in favor of setting certain components to zero.

The second modification is to compute left Jordan chains as well as right Jordan chains, and use them together to eliminate the last s iterations of Algorithm 4.1. Suppose $b_0, \ldots, b_{s-1} \in \mathbb{C}^n$ are given and we wish to find vectors $\xi_0, \ldots, \xi_{s-1}$ such that there exist $\xi_s, \ldots, \xi_{2s-1}$ for which

$$\underbrace{\begin{pmatrix} \mathbb{A}_{s-1} & 0 \\ \mathbb{B}_{s-1} & \mathbb{A}_{s-1} \end{pmatrix}}_{\mathbb{A}_{2s-1}} \begin{pmatrix} \xi_0 \\ \cdots \\ \xi_{2s-1} \end{pmatrix} = \begin{pmatrix} 0; \ldots; 0 \\ b_0; \ldots; b_{s-1} \end{pmatrix}, \quad \mathbb{B}_{s-1} = \begin{pmatrix} A_s & \cdots & A_1 \\ \cdots & \cdots & \cdots \\ A_{2s-1} & \cdots & A_s \end{pmatrix}. \tag{4.15}$$

The first ns equations in (4.15) ensure there is a $w \in \mathbb{C}^d$ such that $(\xi_0; \ldots; \xi_{s-1}) = \mathbb{X}_{s-1} w$. Let $\mathbb{Y}_{s-1}$ be any $d \times ns$ matrix whose rows form a basis for the left nullspace of $\mathbb{A}_{s-1}$. Then applying $\mathbb{Y}_{s-1}$ to the last ns equations in (4.15) gives

$$\mathbb{Y}_{s-1}\mathbb{B}_{s-1}\mathbb{X}_{s-1} w = \mathbb{Y}_{s-1}(b_0; \ldots; b_{s-1}). \tag{4.16}$$

The $d \times d$ matrix $\mathbb{Y}_{s-1}\mathbb{B}_{s-1}\mathbb{X}_{s-1}$ is invertible since $\ker(\mathbb{Y}_{s-1}) = \operatorname{range}(\mathbb{A}_{s-1})$ and $\mathbb{W}_{2s-1} = \iota^s(\mathbb{W}_{s-1})$ implies that $\operatorname{range}(\mathbb{B}_{s-1}\mathbb{X}_{s-1}) \cap \operatorname{range}(\mathbb{A}_{s-1}) = \{0\}$. Thus w and $(\xi_0; \ldots; \xi_{s-1})$ can be determined without computing the other ξ_i. We denote the mapping $(b_0; \ldots; b_{s-1}) \mapsto (\xi_0; \ldots; \xi_{s-1})$ by

$$\text{solve2}(\cdot) := \mathbb{X}_{s-1}(\mathbb{Y}_{s-1}\mathbb{B}_{s-1}\mathbb{X}_{s-1})^{-1}\mathbb{Y}_{s-1}(\cdot). \tag{4.17}$$

The matrix $\mathbb{Y}_{s-1}$ can be obtained by applying Algorithm 3.1 to $A(\lambda)^T$ since

$$(y_k^T, \ldots, y_0^T) \begin{pmatrix} A_0 & 0 & \cdots & 0 \\ A_1 & A_0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ A_k & A_{k-1} & \cdots & A_0 \end{pmatrix} = 0 \quad \Leftrightarrow \quad \begin{pmatrix} A_0^T & 0 & \cdots & 0 \\ A_1^T & A_0^T & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ A_k^T & A_{k-1}^T & \cdots & A_0^T \end{pmatrix} \begin{pmatrix} y_0 \\ \vdots \\ y_k \end{pmatrix} = 0. \tag{4.18}$$

Such vectors $y_0, \ldots, y_k$ are said to form a left Jordan chain for $A(\lambda)$ of length $k+1$ at the origin. Equivalently, the corresponding root function $y(\lambda) = y_0 + \cdots + \lambda^k y_k$ must satisfy $y(\lambda)^T A(\lambda) = O(\lambda^{k+1})$.

ALGORITHM 4.6 (Laurent expansion of $A(\lambda)^{-1}b(\lambda)$, first $q+1$ terms)

determine s, $\mathcal{A}_s$, $\mathbb{X}_{s-1}$, $\mathbb{Y}_{s-1}$ using Algorithm 3.1
if $q < s-1$, increase it to $q = s-1$
$x_0 = \cdots = x_q = 0$
for $k = 0, \ldots, q-s$ (possibly empty loop)
  $(\xi_0; \ldots; \xi_s) = \text{solve } \big[b_k - \sum_{j=1}^{k+s} A_j x_{k+s-j}\big]$, (omit sum if $k = 0$)
  for $i = 0, \ldots, s$
    $x_{k+i} = x_{k+i} + \xi_i$
$(\xi_0; \ldots; \xi_{s-1}) = \text{solve2}\Big[\begin{pmatrix} b_{q-s+1} \\ \cdots \\ b_q \end{pmatrix} - \begin{pmatrix} A_{q+1} & \cdots & A_1 \\ \cdots & \cdots & \cdots \\ A_{q+s} & \cdots & A_s \end{pmatrix} \begin{pmatrix} x_0 \\ \cdots \\ x_q \end{pmatrix} \Big]$
for $i = 1, \ldots, s$
  $x_{q-s+i} = x_{q-s+i} + \xi_{i-1}$
return $(x_0; \ldots; x_q)$

EXAMPLE 4.7 In Example 4.5 with $\mathbb{X}_2$ as in (4.14), we obtain the matrices

$$\frac{1}{2}\begin{pmatrix} -i & -1 & i & -1 & 0 & -1 & -i & 0 & 0 \\ -1 & 0 & -1 & -i & 0 & 0 & 0 & 0 & 0 \\ -i & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} Q_0 & 0 & 0 \\ Q_1 & Q_0 & 0 \\ Q_2 & Q_1 & Q_0 \end{pmatrix} \tag{4.19}$$

for $(\mathbb{Y}_2\mathbb{B}_2\mathbb{X}_2)^{-1}\mathbb{Y}_2$ and $\mathbb{X}_2(\mathbb{Y}_2\mathbb{B}_2\mathbb{X}_2)^{-1}\mathbb{Y}_2$, respectively, with Q_i as in (4.13).

REMARK 4.8 Each column of $\mathbb{X}_{s-1}$ has the form $(0; \ldots; 0; x_0; \ldots; x_k)$ for some $0 \le k \le s-1$. The corresponding column of $\mathbb{B}_{s-1}\mathbb{X}_{s-1}$ contains the coefficients of the truncated residual

$$b(\lambda) = \lambda^{-s}A(\lambda)[\lambda^{s-1-k}(x_0 + \cdots + \lambda^k x_k)] + O(\lambda^s), \tag{4.20}$$

which satisfies $A(\lambda)^{-1}b(\lambda) = \lambda^{-1-k}(x_0 + \cdots + \lambda^k x_k) + O(1)$. See (2.12) above.

4.2 Performance comparison

To evaluate the performance of these algorithms, we generated random matrices A_0 and A_1 with integer entries ranging from 0 to 1000, and random vectors y_1, y_2, y_3 with entries ranging from 0 to 10. For $i = 1, 2, 3$, we zeroed out the last 4 entries of y_i, replaced row $n-i$ of A_0 by $y_i^T A_0$, and set $y_{i,n-i} = -1$, so that $y_i^T A_0 = 0$. We then replaced row n of A_0 by $y_1^T A_1$ to make $y_1 - \lambda(0; \ldots; 0; 1)$ into a left Jordan chain of length 2 for $A(\lambda) = A_0 + \lambda A_1$. The right Jordan chains (which appear as poles in $A(\lambda)^{-1}$) are quite complicated in this construction.

We repeated this process 5 times for each matrix size reported in Figure 1, and recorded the median running time of each algorithm using a 900 MHz Sunblade 2000 with 2GB RAM to compute the first 3 terms ($q = 2$) in the Laurent expansion of $A(\lambda)^{-1}b(\lambda)$, where $b(\lambda) = I_{n\times n}$ or $b(\lambda) = e_1$, as indicated. In all cases encountered, $s = 2$, $p = 3$ and $d = 4$, as expected in the above construction.

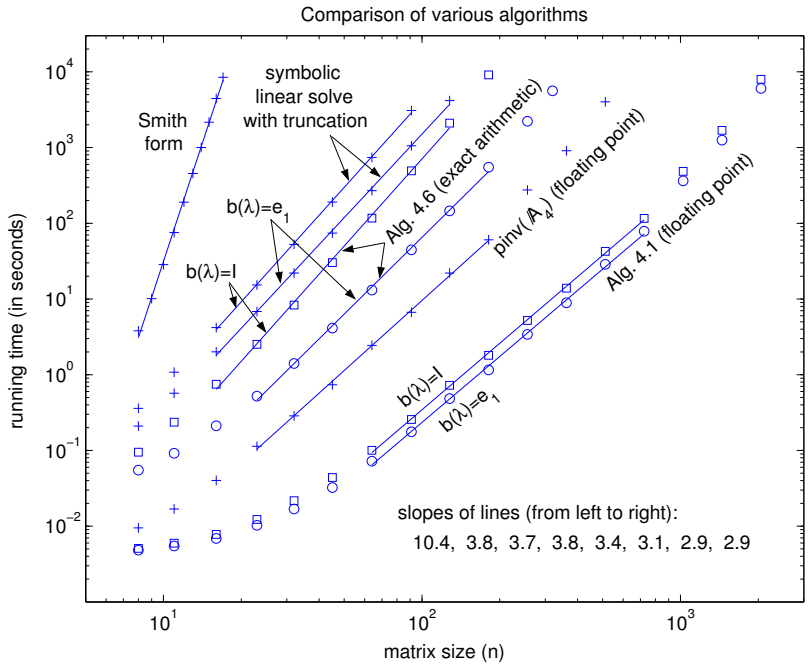

Figure 1: Comparison of various algorithms for computing the Laurent expansion of $(A_0 + \lambda A_1)^{-1} b(\lambda)$ for randomly generated matrix pencils ($q = s = 2$, $d = 4$).

Algorithm 4.1 was implemented in floating point arithmetic using Matlab. Since this method computes the minimum norm solution of $\mathbb{A}_{q+s}(x_0; \ldots; x_{q+s}) = (0; \ldots; 0; b_0; \ldots; b_q)$, we also plot the time it takes Matlab to find the pseudo-inverse of $\mathbb{A}_4$. Algorithm 4.6 was implemented in exact arithmetic using Mathematica. Since the Smith form of $A(\lambda)$ also gives Jordan chain information, we include its running time for comparison (using Maple, which is faster than Mathematica for this). A more competitive benchmark is symbolic factorization with series truncation, which we implemented using Mathematica: each entry of $A(\lambda)$ was given an initial truncation error of $O(\lambda^{q+s+1})$, and each entry of $b(\lambda)$ was given an initial error of $O(\lambda^{q+1})$; Gaussian elimination with complete pivoting (on terms of lowest leading order) was then used to solve the system, truncating series sums and products appropriately. Note that Algorithm 4.6 can solve for n right hand sides faster than the symbolic approach can solve for one right hand side. Also, the symbolic approach requires that s is known a-priori, whereas Algorithm 4.6 does not.

The best-fit lines in Figure 1 have been placed in regimes that are large enough that lower order terms in the operation count become negligible, but small enough that the computer hardware can operate efficiently. The slopes of the lines indicate the scaling exponent α in $t(n) = Cn^\alpha$. It appears that Algorithm 4.6 scales better when $b(\lambda) = e_1$ than when $b(\lambda) = I$. This is presumably because the rational numbers involved in the LU decomposition of $\mathcal{A}_2$ do not reach their full complexity

until the final stages of the factorization, whereas for $b(\lambda) = I$, all $O(n^3)$ operations of the back-solve stage involve these large rational numbers. We also see that the floating point algorithm scales as $O(n^3)$, as predicted by a straightforward floating point operation count. In many cases where $A(0)$ has special structure, it should be possible to make use of the fact that $\mathcal{A}_s$ contains $A(0)$ as a principal submatrix to further improve this scaling.

A Numerical stability in floating point arithmetic

In this section we will show that the algorithms presented in this paper are forward stable in floating point arithmetic with a condition number that depends on how close $A(\lambda)$ is to an even more singular analytic matrix function. This is possible because we have specified the location of the eigenvalue of interest (e.g. the origin) as part of the problem statement. Thus, we have to know more about the matrix than can be determined by looking at a numerical approximation of it — we have to know that $A(0)$ is exactly singular. In particular, we do not claim to be able to compute the Jordan canonical form of a matrix in a stable way unless the eigenvalues are known a-priori, or can be determined algebraically (using exact arithmetic for that part of the calculation). If the multiplicities can be determined algebraically, the eigenvalues can be computed in floating point arithmetic (using extra precision if necessary), but there is no way to determine affirmatively that a cluster of eigenvalues should coalesce using floating point arithmetic alone.

Finding the Jordan chains of an analytic matrix function *at the origin* is a generalization of the familiar problem of finding the kernel of an $n \times n$ matrix A, which is solved numerically as follows. The user supplies a numerical approximation $\tilde{A}$ of A and a tolerance σ_{thresh}. The singular value decomposition $\tilde{A} = U\Sigma V^*$ is then computed and the right singular vectors corresponding to singular values $\sigma_j < \sigma_{\text{thresh}}$ are returned as a basis for the kernel. This procedure is forward stable due to the backward stability of the SVD algorithm:

THEOREM A.1 *(Backward stability of the SVD). Let $\tilde{A}$ be a floating point representable $n \times n$ matrix. There is a (small) constant C independent of $\tilde{A}$ (and n) such that the result of computing the SVD of $\tilde{A}$ in floating point arithmetic (using Givens rotations for the bi-diagonal reduction process) is the exact SVD of a nearby matrix $\hat{A}$ satisfying*

$$\hat{A} = \hat{U}\hat{\Sigma}\hat{V}^*, \qquad \|\hat{A} - \tilde{A}\| \leq Cn^2\varepsilon\|\tilde{A}\|, \tag{A.1}$$

where ε is machine precision and the matrix multiplication is in exact arithmetic.

REMARK A.2 The matrices $\hat{U}$ and $\hat{V}$ are stored implicitly as compositions of Givens rotations so they are exactly orthogonal. If Householder reflections are used in the reduction to bidiagonal form, the bound on the right hand side of (A.1) involves the Frobenius norm instead of the 2-norm. This would require minor modifications of the analysis below.

THEOREM A.3 *(Forward stability of computing nullspaces). Suppose* $A = U\Sigma V^*$ *is an* $n \times n$ *matrix with* $\dim\ker A = p \geq 0$. *Let* $\text{gap} = \sigma_{n-p}$ *be the smallest non-zero singular value of* A, *and suppose a tolerance* σ_{thresh} *is given such that*

$$\sigma_{thresh} \leq \frac{1}{3}\,\text{gap}\,. \tag{A.2}$$

Let $\alpha \geq \|A\|$, $\gamma > 0$, *and* $\tilde{A}$ *be any floating point representable matrix satisfying*

$$\|\tilde{A} - A\| \leq \gamma\alpha n^2\varepsilon, \qquad \gamma n^2\varepsilon \leq 1/3. \tag{A.3}$$

Let $\hat{A} = \hat{U}\hat{\Sigma}\hat{V}^*$ *be the result of computing the SVD of* $\tilde{A}$ *in floating point arithmetic. Then*

$$\hat{A} = A + E, \qquad \|E\| \leq \left(\gamma + \frac{4}{3}C\right)\alpha n^2\varepsilon, \qquad |\hat{\sigma}_j - \sigma_j| \leq \|E\|, \quad (1 \leq j \leq n), \tag{A.4}$$

where C *is the backward stability constant of the SVD algorithm from (A.1). In particular, if*

$$\varepsilon < \frac{\sigma_{thresh}}{(\gamma + 4C/3)\alpha n^2}, \tag{A.5}$$

the number of singular values satisfying $\hat{\sigma}_j < \sigma_{thresh}$ *is* p *and there is an orthogonal* $n \times p$ *nullspace matrix* X *for* A *such that*

$$\|\tilde{X} - X\| \leq \frac{1.1\|E\|}{\text{gap}}, \qquad AX = 0, \qquad X^*X = I_{p\times p}, \tag{A.6}$$

where $\tilde{X}$ *is the numerical nullspace matrix consisting of the last* p *columns of* V *(stored implicitly using Givens rotations). Finally, if we explicitly compute the matrix entries of* $\tilde{X}$ *in floating point arithmetic, we obtain a matrix* $\hat{X}$ *satisfying* $\|\hat{X} - \tilde{X}\| \leq \frac{1}{2}Cn^2\varepsilon$ *as well as*

$$\|\hat{X} - X\| \leq K\varepsilon, \qquad K = (1.1\gamma + 2C)n^2\frac{\alpha}{\text{gap}}. \tag{A.7}$$

REMARK A.4 The bound (A.4) on the singular values follows from Weyl's theorem [7]. The bound (A.6) on $\tilde{X}$ follows from the "$\sin 2\Theta$" theorem [6] on the perturbation of invariant subspaces. The constant 1.1 in (A.6) is related to the choice of 1/3 in (A.2), for if $0 \leq \theta \leq \frac{\pi}{4}$ and $\frac{1}{2}\sin 2\theta \leq \frac{1}{3}$ then $2\sin\frac{\theta}{2} \leq \frac{1.1}{2}\sin 2\theta$. The requirement $\gamma n^2\varepsilon \leq 1/3$ is satisfied automatically if (A.2) and (A.5) hold (since $\text{gap} \leq \alpha$). The constant α can be any convenient bound on $\|A\|$ (including $\|A\|$ itself). The purpose of γ is to quantify the effect of starting with a poor approximation $\tilde{A}$ of A when analyzing Algorithm 3.1 below.

In summary, given A and and a tolerance σ_{thresh} satisfying (A.2), the nullspace matrix $\hat{X}$ computed in floating point arithmetic is close to an exact nullspace matrix X as long as ε satisfies (A.5), and the error can be made arbitrarily small by taking $\varepsilon \to 0$. Thus, computing nullspaces is *forward stable* in floating point

arithmetic with condition number K proportional to $n^2\|A\|/\operatorname{gap}$. Note that any perturbation of A that decreases the dimension of the kernel will have a much larger condition number due to the small, non-zero singular value(s) created by the perturbation.

Except for the last step where we explicitly compute the matrix entries of $\hat{X}$, this procedure for computing the nullspace is also *backward stable* as zeroing out the p smallest singular values $\hat{\sigma}_j$ will not change $\hat{A}$ very much; however, backward stability is less useful than forward stability when the underlying problem depends discontinuously on the matrix entries. For example, the algorithm "return the $n \times 0$ empty matrix X" is also backward stable as there are invertible matrices arbitrarily close to any singular matrix.

Before analyzing the stability of Algorithm 3.1, it is instructive to consider the naive algorithm employing the SVD to compute the nullspace matrices $\mathbb{X}_k$ directly from $\mathbb{A}_k$ for $k = 0, 1, 2, \ldots$, until $R_k = \dim\ker \mathbb{A}_k$ stops increasing. We identify the maximal Jordan chain length s as the first k for which $R_k = R_{k-1}$ (with R_{-1} taken to be 0). The matrix $\mathbb{X}_{s-1}$ then contains all the Jordan chain information for $A(\lambda)$. Setting $\gamma = 1$ and $\alpha = \|\mathbb{A}_{s-1}\|$ in (A.7), we see that a natural candidate for the condition number of this problem is

$$K_{\text{naive}} = \frac{(1.1 + 2C)s^2n^2\|\mathbb{A}_{s-1}\|}{\operatorname{gap}_{s-1}^{\text{naive}}}, \tag{A.8}$$

where $\operatorname{gap}_k^{\text{naive}}$ is the smallest non-zero singular value of $\mathbb{A}_k$. The bounds in (A.4) above imply that given $A(\lambda)$ of maximal Jordan chain length s at the origin and a tolerance

$$\sigma_{\text{thresh}} \;\le\; \min_{0\le k\le s}\left(\frac{1}{3}\operatorname{gap}_k^{\text{naive}}\right), \tag{A.9}$$

the naive algorithm will compute the dimensions $R_0, \ldots, R_s$ correctly as long as

$$\varepsilon < \frac{\sigma_{\text{thresh}}}{(1 + 4C/3)(s+1)^2n^2\|\mathbb{A}_s\|} \tag{A.10}$$

and as long as our numerical approximations $\tilde{A}_k$ of the coefficients of $A(\lambda)$ lead to augmented matrices that satisfy $\|\tilde{\mathbb{A}}_k - \mathbb{A}_k\| \le n^2(k+1)^2\varepsilon\|\mathbb{A}_k\|$ for $0 \le k \le s$. Moreover, the computed nullspace $\hat{\mathbb{X}}_{s-1}$ will be close to an exact nullspace matrix $\mathbb{X}_{s-1}$ for $\mathbb{A}_{s-1}$:

$$\|\hat{\mathbb{X}}_{s-1} - \mathbb{X}_{s-1}\| \le K_{\text{naive}}\,\varepsilon. \tag{A.11}$$

Thus, the naive algorithm is forward stable with condition number K_{naive}. However, this analysis overlooks one important detail: computing the nullspace of $\mathbb{A}_{s-1}$ using the SVD will not respect the Jordan chain structure of this nullspace; hence, there are likely to be internal inconsistencies when we try to extract Jordan chains from $\mathbb{X}_{s-1}$. This is illustrated in the following two examples:

EXAMPLE A.5 Let $A(\lambda) = P\begin{pmatrix} 1+\lambda & 0 & 0 \\ 0 & a & 0 \\ 3\lambda & 0 & \lambda^2 \end{pmatrix} Q^*$, where P and Q are randomly generated orthogonal matrices and $a = 10^{-8}$. (The purpose of P and Q is to

cause roundoff error). We find that $s = 2$ and $\mathrm{gap}_k^{\mathrm{naive}} = a$ for $0 \le k \le 2$. When we compute $\mathbb{X}_1$ using the SVD, we obtain the kernel $\hat{\mathbb{X}}_1$ of a nearby matrix $\hat{\mathbb{A}}_1$ with the same nullspace dimension as $\mathbb{A}_1$. (We ignore the errors in computing the matrix entries of $\hat{\mathbb{X}}_1$ from the Givens rotations). One possibility would be

$$\hat{\mathbb{A}}_1 = \begin{pmatrix} P & 0 \\ 0 & P \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & & & \\ 0 & a & \varepsilon & & & \\ 0 & 0 & 0 & & & \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & a & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} Q^* & 0 \\ 0 & Q^* \end{pmatrix}, \quad \hat{\mathbb{X}}_1 = \begin{pmatrix} Q & 0 \\ 0 & Q \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & -\varepsilon/a \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{pmatrix}.$$

If we denote the second column of $\hat{\mathbb{X}}_1$ by $(x_0; x_1)$, we see that the first column of $\hat{\mathbb{X}}_1$ deviates from the linear span of $\{(x_0; x_1),\ (0; x_0)\}$ by approximately ε/a, which is of the same order as a in double precision arithmetic. Thus, if we wish to post-process $\hat{\mathbb{X}}_1$ to obtain a consistent Jordan chain structure (using e.g. the SVD with a threshold to coalesce subspaces), we need the right hand side of (A.11) to satisfy

$$K_{\mathrm{naive}}\, \varepsilon < \sigma_{\mathrm{thresh}}. \tag{A.12}$$

This introduces an additional factor of $\mathrm{gap}_{s-1}^{\mathrm{naive}}$ in the requirement (A.10). In the current example, this pushes the requirement beyond double precision arithmetic.

EXAMPLE A.6 Let $A(\lambda) = P \begin{pmatrix} 1+\lambda & 0 & 0 \\ 0 & a+\lambda & 0 \\ 3\lambda & 0 & \lambda^2 \end{pmatrix} Q^*$, where $a = 10^{-5}$. We again find that $s = 2$, but this time $\mathrm{gap}_0^{\mathrm{naive}} = a$, $\mathrm{gap}_1^{\mathrm{naive}} \approx a^2$, and $\mathrm{gap}_2^{\mathrm{naive}} \approx a^3$. If we use double precision arithmetic, the singular vector corresponding to $\mathrm{gap}_2^{\mathrm{naive}}$ will likely be interpreted incorrectly as an element of the kernel of $\mathbb{A}_2$. Since the next gap in the list of singular values of $\mathbb{A}_2$ is very large (around $0.28 \gg \varepsilon$), the linear span of these singular vectors will be computed very accurately and the naive algorithm will return

$$\hat{\mathbb{X}}_2^* = Z \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & a^2/c & 0 & 0 & -a/c & 0 & 0 & 1/c & 0 \end{pmatrix} \begin{pmatrix} Q^* & 0 & 0 \\ 0 & Q^* & 0 \\ 0 & 0 & Q^* \end{pmatrix} + O(\varepsilon), \tag{A.13}$$

where $c = \sqrt{1 + a^2 + a^4}$ and Z is an orthogonal 3×3 matrix. This matrix $\hat{\mathbb{X}}_2$ is incompatible with any Jordan chain structure since the chain of length 3 is not an extension of a shorter chain. This shows the importance of the requirement (A.9) on the success of the naive algorithm.

Algorithm 3.1 avoids internal inconsistencies in the nullspace matrices $\hat{\mathbb{X}}_k$ by constructing them incrementally, with the Jordan structure built in. The price we pay for this is that we do not look ahead at A_{k+1} when computing $\mathbb{X}_k$ to help guide the choice of insignificant digits (lost to roundoff error) in order to line up the

subspaces at the next iteration (to find chains of length $k+1$). Thus, Algorithm 3.1 may require more precision than the naive algorithm to find long chains, but the naive algorithm will require additional precision to make sense of the nullspaces it finds (and to avoid finding too many null vectors).

Examples A.5 and A.6 above are cases where the naive algorithm fails in double precision arithmetic while Algorithm 3.1 succeeds (with typically 9 correct digits in $\hat{\mathbb{X}}$ in the former case and 7 in the latter). If we set $a = 10^{-8}$ and replace the second column of $A(\lambda)$ by $(0; a; \lambda)$, then the naive algorithm will succeed in double precision arithmetic while Algorithm 3.1 will fail to find the chain of length 2. If we instead replace the second column by $(0; a + \lambda; \lambda)$, then both algorithms will fail unless extra precision is used: the naive algorithm will find a spurious chain of length 3 while Algorithm 3.1 will fail to find the chain of length 2. In all cases, both algorithms will succeed once ε is reduced to satisfy (A.12) or (A.28) below.

These examples can be understood by noting that the error in computing $\ker A_0$ is overwhelmingly in the direction of the second right singular vector $v_2 = Q(:, 2)$ due to the large gap between σ_1 and σ_2 and the small gap between σ_2 and $\sigma_3 = 0$. The requirements on ε for the success of the algorithms (and in particular the quantities $\mathrm{gap}_k^{\mathrm{naive}}$) depend on where A_1 maps v_2. The examples above illustrate several of the possibilities.

We now prove that Algorithm 3.1 is *forward stable* in floating point arithmetic. Suppose $A(\lambda)$ is exactly singular at the origin with maximal Jordan chain length s. We define

$$\alpha_0 = \|A_0\|, \tag{A.14}$$

$$\alpha_k = \left(1 + \|A_0\|^2 + \cdots + \|A_k\|^2\right)^{1/2}, \qquad (0 \le k \le s), \tag{A.15}$$

$$\sigma_j^{(k)} = \ j\text{th singular value of } \mathcal{A}_k, \qquad \left(\sigma_1^{(k)} \ge \cdots \ge \sigma_{n+R_{k-1}}^{(k)}\right) \tag{A.16}$$

$$\mathrm{gap}_k = \sigma_{n+R_{k-1}-r_k}^{(k)} = \ \text{smallest non-zero singular value of } \mathcal{A}_k. \tag{A.17}$$

The quantities α_k are convenient upper bounds for $\|\mathcal{A}_k\|$, and, as before, $R_{-1} = 0$. Note that $\sigma_j^{(k)}$ and gap_k are independent of which orthogonal matrices $\mathbb{X}_k$ and X_k we use to represent the kernels. Also note that gap_s is the smallest singular value of $\mathcal{A}_s$, which is invertible. The user must supply a threshold σ_{thresh} for computing nullspaces that satisfies

$$\sigma_{\mathrm{thresh}} \le \frac{1}{3}\left(\min_{0\le k\le s} \mathrm{gap}_k\right). \tag{A.18}$$

This requirement is often much less strict than (A.9). We assume that numerical approximations of $A_0, \ldots, A_s$ are known to a tolerance

$$\|\tilde{A}_k - A_k\| \le n^2 \varepsilon \, \|A_k\|, \qquad (0 \le k \le s). \tag{A.19}$$

The first step of the algorithm is to compute the kernel $\mathbb{X}_0$ of A_0 as described above. In light of (A.19) and (A.14), Theorem A.3 implies that $\hat{\mathbb{X}}_0$ satisfies

$$\|\hat{\mathbb{X}}_0 - \mathbb{X}_0\| \le (1.1 + 2C) n^2 \frac{\alpha_0}{\mathrm{gap}_0}\varepsilon, \qquad \left(\varepsilon < \frac{\sigma_{\mathrm{thresh}}}{(1 + 4C/3)\alpha_0 \, n^2}\right). \tag{A.20}$$

Next we choose $\delta > 0$ (to be determined later) and formulate the induction hypothesis

$$\|\hat{\mathbb{X}}_k - \mathbb{X}_k\| \le C_k(n + R_{k-1})^2\varepsilon \le \delta. \tag{A.21}$$

From (A.20), we know $C_0 = (1.1 + 2C)\alpha_0/\operatorname{gap}_0$ works when $k = 0$. Our goal is to derive a recursion for C_k that ensures that s is computed correctly, and that (A.21) holds for $1 \le k \le s-1$ as well. To this end, we compute

$$\tilde{\mathcal{A}}_{k+1} = \operatorname{fl}\left[\begin{pmatrix} \tilde{A}_{k+1} & \tilde{A}_k & \cdots & \tilde{A}_0 \\ 0 & (& \hat{\mathbb{X}}_k^* &) \end{pmatrix} \begin{pmatrix} 0 & \hat{\mathbb{X}}_k \\ I_{n\times n} & 0 \end{pmatrix} \right], \tag{A.22}$$

where $\operatorname{fl}(\cdot)$ indicates that the calculation is done in floating point arithmetic, and use (A.21) to conclude

$$\|\tilde{\mathcal{A}}_{k+1} - \mathcal{A}_{k+1}\| \le \gamma_k\alpha_{k+1}(n + R_k)^2\varepsilon, \qquad \gamma_k = (1+\delta)^2[(k+2) + 2C_k]. \tag{A.23}$$

Next we compute $[\hat{V}_k; \hat{U}_k] = \operatorname{fl}\left(\ker \tilde{\mathcal{A}}_{k+1}\right)$ and use Theorem A.3 to determine

$$\left\|[\hat{V}_k; \hat{U}_k] - [V_k; U_k]\right\| \le (1.1\gamma_k + 2C)\,\frac{\alpha_{k+1}}{\operatorname{gap}_{k+1}}(n + R_k)^2\varepsilon. \tag{A.24}$$

Finally, we compute

$$\hat{X}_{k+1} = \operatorname{fl}\left[\begin{pmatrix} 0 & \hat{\mathbb{X}}_k \\ I & 0 \end{pmatrix} \begin{pmatrix} \hat{V}_k \\ \hat{U}_k \end{pmatrix} \right], \qquad \hat{\mathbb{X}}_{k+1} = \left[\begin{pmatrix} 0 \\ \mathbb{X}_k \end{pmatrix}, \begin{pmatrix} X_{k+1} \end{pmatrix} \right] \tag{A.25}$$

to obtain

$$\left\|\hat{\mathbb{X}}_{k+1} - \mathbb{X}_{k+1}\right\| \le \underbrace{\left((2+\delta)C_k + (1+\delta)^2 + \left[1.1\gamma_k + 2C\right]\frac{\alpha_{k+1}}{\operatorname{gap}_{k+1}}\right)}_{C_{k+1}}(n + R_k)^2\varepsilon.$$

We now select $\delta = 0.1$ and use the fact that $\alpha_{k+1} \ge \operatorname{gap}_{k+1}$ to obtain a simpler recursion (with larger terms):

$$C_{k+1} = \left(5C_k + \frac{4}{3}(k+3) + 2C\right)\frac{\alpha_{k+1}}{\operatorname{gap}_{k+1}}, \qquad C_0 = (1.1 + 2C)\frac{\alpha_0}{\operatorname{gap}_0}. \tag{A.26}$$

This recursion can be solved explicitly:

$$C_k = \left[\frac{5^{k+1} - 1}{2}C + \frac{131}{60}5^k - \frac{13}{12} - \frac{k}{3}\right]\frac{\alpha_k}{\operatorname{gap}_k}\cdots\frac{\alpha_0}{\operatorname{gap}_0}, \tag{A.27}$$

which yields γ_k as well. In order for this analysis to be valid (and to determine that $k = s$ is the first index for which $\tilde{\mathcal{A}}_k$ is invertible), we require

$$\varepsilon < \min\left(\frac{\delta}{C_{s-1}(n + R_{s-2})^2}\,,\ \frac{\sigma_{\text{thresh}}}{(\gamma_{s-1} + 4C/3)\alpha_s(n + R_{s-1})^2}\right). \tag{A.28}$$

The bounds on $\|\hat{U}_k - U_k\|$ and $\|\hat{V}_k - V_k\|$ in (A.24) can now be used to derive error estimates for the extracted Jordan chains $\widetilde{X}_k$ in Algorithm 3.1. The bound (A.23) on $\|\tilde{\mathcal{A}}_s - \mathcal{A}_s\|$ ensures that Algorithm 4.1 is forward stable in floating point arithmetic once $b(\lambda)$ and the number of desired terms q in the Laurent expansion of $A(\lambda)^{-1}b(\lambda)$ have been specified.

REMARK A.7 The exponential dependence of C_k on k in (A.28) is not likely to cause difficulty as problems with Jordan chain lengths longer than $s = 3$ are very rare in practice. Of more concern is the fact that the ratios $\alpha_k/\operatorname{gap}_k$ *multiply* from one iteration to the next. Example A.6 above shows that this should be expected from a general (i.e. worst case) analysis. In that example, the quantities gap_k are all equal to a while $\operatorname{gap}_k^{\text{naive}} \approx a^{k+1}$ for $0 \le k \le 2$. Example A.5 shows that sometimes Algorithm 3.1 will perform better than predicted by this worst case analysis — it depends how the errors in $\hat{\mathbb{X}}_k$ propagate through the coefficients $\tilde{A}_j$ when $\tilde{\mathcal{A}}_{k+1}$ is formed in (A.22). However, a general analysis that keeps track of the directions of the errors as well as their norms would be difficult.

In summary, the error in the quantities computed in Algorithms 3.1 and 4.1 using floating point arithmetic can be made arbitrarily small (relative to their nearest counterparts in exact arithmetic) by taking $\varepsilon \to 0$. Moreover, the convergence rates are linear in ε with constants that depend on the distance to an even more singular analytic matrix function (i.e. on the sizes of the ratios $\frac{\alpha_k}{\operatorname{gap}_k}$, $0 \le k \le s-1$):

$$\|\hat{\mathbb{X}}_{s-1} - \mathbb{X}_{s-1}\| \le K\varepsilon, \qquad K = C_{s-1}(n + R_{s-2})^2. \tag{A.29}$$

The convergence is linear because we have specified the location of the singularity (the origin) in advance. In some problems, the zeros of $\Delta(\lambda) = \det A(\lambda)$ must also be found, in which case we recommend Davies' method [5] for locating the zeros of an analytic function using contour integration techniques. But if $\Delta(\lambda)$ has a zero of order d at λ_0, roundoff error is likely to split this zero like the dth root of ε rather than linearly, making these zeros difficult to compute.

It is often possible to use the methods of this paper to stabilize the problem of finding these zeros. For example, in the corner singularities problem described in the introduction, it is very common for $\Delta(\lambda)$ to have zeros at the integers; the corresponding singular functions are polynomials. If another zero λ_1 of $\Delta(\lambda)$ is close to such an integer λ_0, Davies' method will likely give inaccurate results as the zeros will interfere with each other. But if we first compute a canonical system of Jordan chains at the integer λ_0, we learn the order d of the zero of $\Delta(\lambda)$ at λ_0 using an $O(\varepsilon)$ method; we can then find λ_1 using Davies' method on $\Delta(\lambda)(\lambda - \lambda_0)^{-d}$, which also gives $O(\varepsilon)$ accuracy if λ_1 is a simple zero. In this way we can find parasitic zeros near a known (possibly multiple) zero with high accuracy.

References

[1] K. E. Avrachenkov, M. Haviv, and P. G. Howlett. Inversion of analytic matrix functions that are singular at the origin. *SIAM J. Matrix Anal. Appl.*, 22(4):1175–1189, 2001.

[2] H. Baumgärtel. *Analytic Perturbation Theory for Matrices and Operators.* Birkhäuser, Basel, 1985.

[3] S. L. Campbell. *Singular systems of differential equations II*, volume 61 of *Research Notes in Mathematics*. Pitman, London, 1982.

[4] Martin Costabel and Monique Dauge. Construction of corner singularities for Agmon-Douglis-Nirenberg elliptic systems. *Math. Nachr.*, 162:209–237, 1993.

[5] B. Davies. Locating the zeros of an analytic function. *J. Comput. Phys.*, 66(1):36–49, 1986.

[6] C. Davis and W. M. Kahan. The rotation of eigenvectors by a perturbation. III. *SIAM J. Numer. Anal.*, 7(1):1–46, 1970.

[7] James W. Demmel. *Applied Numerical Linear Algebra*. SIAM, Philadelphia, 1997.

[8] A. Edelman, E. Elmroth, and B. Kågström. A geometric approach to perturbation theory of matrices and matrix pencils, part I: versal deformations. *SIAM J. Matrix Anal. Appl.*, 18(3):653–692, 1997.

[9] I. Gohberg, M. A. Kaashoek, and F. van Schagen. On the local theory of regular analytic matrix functions. *Linear Algebra Appl.*, 182:9–25, 1993.

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

[11] M. Haviv and Y. Ritov. On series expansions and stochastic matrices. *SIAM J. Matrix Anal. Appl.*, 14(3):670–676, 1993.

[12] M. V. Keldysh. On the characteristic values and characteristic functions of certain classes of non-selfadjoint equations. *Dokl. Akad. Nauk USSR (Russian)*, 77:11–14, 1951. English transl. in [15].

[13] P. Lancaster. Inversion of lambda-matrices and application to the theory of linear vibrations. *Arch. Rational Mech. Anal.*, 6(2):105–114, 1960.

[14] C. E. Langenhop. The Laurent expansion for a nearly singular matrix. *Linear Algebra Appl.*, 4:329–340, 1971.

[15] A. S. Markus. *Introduction to the Spectral Theory of Polynomial Operator Pencils*, volume 71 of *Translations of Mathematical Monographs*. Americal Mathematical Society, Providence, 1988.

[16] M. K. Sain and J. L. Massey. Invertibility of linear time-invariant dynamical systems. *IEEE Trans. Automat. Control*, (2):141–149, 1969.

[17] P. Schweitzer and G. W. Stewart. The Laurent expansion of pencils that are singular at the origin. *Linear Algebra Appl.*, 183:237–254, 1993.