Computing Krylov iterates in the time of matrix multiplication

Vincent Neiger
Sorbonne Université, CNRS, LIP6
F-75005 Paris, France

Clément Pernet
Univ. Grenoble Alpes, Grenoble INP
CNRS, LJK UMR 5224
Grenoble, France

Gilles Villard
CNRS, Univ. Lyon, ENS de Lyon, Inria,
UCBL, LIP UMR 5668
Lyon, France

ABSTRACT

Krylov methods rely on iterated matrix-vector products $A^k u_j$ for an $n \times n$ matrix A and vectors $u_1, \ldots, u_m$. The space spanned by all iterates $A^k u_j$ admits a particular basis — the *maximal Krylov basis* — which consists of iterates of the first vector $u_1, Au_1, A^2u_1, \ldots$, until reaching linear dependency, then iterating similarly the subsequent vectors until a basis is obtained. Finding minimal polynomials and Frobenius normal forms is closely related to computing maximal Krylov bases. The fastest way to produce these bases was, until this paper, Keller-Gehrig's 1985 algorithm whose complexity bound $O(n^\omega \log(n))$ comes from repeated squarings of A and logarithmically many Gaussian eliminations. Here $\omega > 2$ is a feasible exponent for matrix multiplication over the base field. We present an algorithm computing the maximal Krylov basis in $O(n^\omega \operatorname{loglog}(n))$ field operations when $m \in O(n)$, and even $O(n^\omega)$ as soon as $m \in O(n/\log(n)^c)$ for some fixed real $c > 0$. As a consequence, we show that the Frobenius normal form together with a transformation matrix can be computed deterministically in $O(n^\omega(\operatorname{loglog}(n))^2)$, and therefore matrix exponentiation A^k can be performed in the latter complexity if $\log(k) \in O(n^{\omega-1-\varepsilon})$ for some fixed $\varepsilon > 0$. A key idea for these improvements is to rely on fast algorithms for $m \times m$ polynomial matrices of average degree n/m, involving high-order lifting and minimal kernel bases.

KEYWORDS

Krylov iteration; Frobenius normal form; Polynomial linear algebra.

ACM Reference Format:
Vincent Neiger, Clément Pernet, and Gilles Villard. 2024. Computing Krylov iterates in the time of matrix multiplication. In *International Symposium on Symbolic and Algebraic Computation (ISSAC '24), July 16–19, 2024, Raleigh, NC, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3666000.3669715

1 INTRODUCTION

We present a new deterministic algorithm for the computation of some specific Krylov matrices, which play a central role in determining the structure of linear operators. To a matrix $A \in \mathbb{F}^{n\times n}$ over an arbitrary commutative field $\mathbb{F}$, a (column) vector $u \in \mathbb{F}^n$ and a nonnegative integer $d \in \mathbb{N}$, we associate the *Krylov matrix* $\mathrm{K}_d(A, u)$ formed by the first d iterates of u through A:

$$\mathrm{K}_d(A,u) = \begin{bmatrix} u & Au & \cdots & A^{d-1}u \end{bmatrix} \in \mathbb{F}^{n\times d}. \qquad (1)$$

More generally, to m vectors $U = \begin{bmatrix} u_1 & \cdots & u_m \end{bmatrix} \in \mathbb{F}^{n\times m}$ and a tuple $d = (d_1, \ldots, d_m) \in \mathbb{N}^m$, we associate the Krylov matrix

$$\mathrm{K}_d(A,U) = \begin{bmatrix} \mathrm{K}_{d_1}(A,u_1) & \cdots & \mathrm{K}_{d_m}(A,u_m) \end{bmatrix} \in \mathbb{F}^{n\times |d|}, \qquad (2)$$

where $|d|$ is the sum $d_1 + \cdots + d_m$. (Note that it will prove convenient to allow $m = 0$ and $d_i = 0$.) Such matrices are used to construct special bases of the A-invariant subspace

$$\mathrm{Orb}(A,U) = \mathrm{Span}_{\mathbb{F}}(\{A^i u_j, i \in \mathbb{N}, j \in \{1,\ldots,m\}\}).$$

Indeed, for a given A and U there always exists a tuple d such that the columns of $\mathrm{K}_d(A, U)$ form a basis of $\mathrm{Orb}(A, U)$ (Section 3.1.1). In this paper, we focus on the computation of the unique such basis of $\mathrm{Orb}(A, U)$ whose tuple d is the lexicographically largest one [12, Sec. 5]. We call this d the *maximal (Krylov) indices* of $\mathrm{Orb}(A, U)$, and the corresponding basis the *maximal Krylov basis*.

Our main result is a deterministic algorithm that computes the maximal Krylov basis in $O(n^\omega \operatorname{loglog}(n))$ field operations when $m \in O(n)$, where $\omega > 2$ is a feasible exponent for the cost of square matrix multiplication over $\mathbb{F}$ [1, 3, 24]. The bound becomes $O(n^\omega)$ as soon as the number m of initial vectors in U is in $O(n/\log(n)^c)$ for some constant $c > 0$ (see Theorem 4.1). This is an improvement over the best previously known complexity bound $O(n^\omega \log(n))$, for an algorithm due to Keller-Gehrig [12]. In particular, to get down to $O(n^\omega)$, we avoid an ingredient that is central in the latter algorithm and related ones, which is to compute logarithmically many powers of A by repeated squaring (see Section 3.1.1).

Overview of the approach. The main idea is to use operations on polynomial matrices rather than linear transformations. In this direction, we are following in the footsteps of e.g. [27], where polynomial matrix inversion is exploited to compute sequences of matrix powers, and [16], where polynomial matrix normal forms and block-triangular decompositions allow the efficient computation of the characteristic polynomial. A key stage we introduce consists in transforming between left and right matrix fraction descriptions:

$$S(x)T(x)^{-1} = (I - xA)^{-1}U = \sum_{k\geq 0} x^k A^k U, \qquad (3)$$

with $S \in \mathbb{F}[x]^{n\times m}$ and $T \in \mathbb{F}[x]^{m\times m}$. For $m \leq n$, one may see Eq. (3) as considering a compressed description ST^{-1}, which has larger polynomial degrees and smaller matrix dimensions than the description $(I - xA)^{-1}U$. The power series expansion of ST^{-1}, when suitably truncated, produces a Krylov basis.

After some preliminary reminders on polynomial matrices in Section 2, the first algorithms are given in Section 3. There, our contribution is specifically adapted to the case where the compression

The authors are supported by Agence nationale de la recherche (ANR) projects ANR-23-CE48-0003-01 CREAM, ANR-22-PECY-0010-01 PEPR Cryptanalyse and ANR-21-CE39-0006 SANGRIA; by the joint ANR-Austrian Science Fund FWF projects ANR-22-CE91-0007 EAGLES and by the EOARD-AFOSR project FA8665-20-1-7029.

ISSAC '24, July 16–19, 2024, Raleigh, NC, USA

ACM ISBN 979-8-4007-0696-7/24/07
https://doi.org/10.1145/3666000.3669715

is fully effective, that is, when m is away from n (at least slightly: $O(n/\log(n)^c)$ suffices). In this case, appropriate S and T are computed using $O(n^\omega)$ arithmetic operations thanks to the kernel basis algorithm of [26] and its analysis in [9, 16] (Section 2.1).

The next steps are to determine the maximal indices and to compute a truncated series expansion of ST^{-1}. First we explain in Section 3.1 that the maximal indices of $\mathrm{Orb}(A, U)$ are obtained by working, equivalently to Eq. (3), from certain denominator matrices of $(xI - A)^{-1}U$ (see Eq. (4)). The indices are computed as diagonal degrees if the denominator is triangular (Lemma 3.1). It follows that a Hermite form computation allows us to obtain them efficiently [13]. We then give in Section 3.2 an algorithm for computing a Krylov matrix $\mathrm{K}_d(A, U)$ for an arbitrary given d, which we will apply afterwards with the maximal indices for d. According to Eq. (3) and given a tuple $d = (d_1, \ldots, d_m) \in \mathbb{N}^m$, this Krylov matrix can be obtained from the expansion of ST^{-1} with column j truncated modulo x^{d_j} for $1 \le j \le m$. To deal with the unbalancedness of degrees and truncation order, this expansion is essentially computed using high-order lifting [22], combined with the partial linearization technique of [7]. (To avoid diverting the focus of Section 3, details about this are deferred until Section 6.) If $m \in O(n/\log(n)^c)$, both the maximal indices and $\mathrm{K}_d(A, U)$ for any d such that $|d| = O(n)$ can be computed using $O(n^\omega)$ arithmetic operations.

Our general algorithm computing maximal Krylov bases is given in Section 4. The ability to reduce the cost for certain m, as seen above, allows to improve the general case $m = O(n)$ and obtain the complexity bound $O(n^\omega \operatorname{loglog}(n))$. Algorithm `MaxKrylovBasis` is a hybrid one, using the Keller-Gehrig strategy as well as polynomial matrices. In the same spirit as the approach in [12, Sec. 5], we start with the partial computation of a maximal Krylov basis, from the whole U but for only a few iterations, i.e. only considering iterates $A^{k_j}u_j$ for k_j bounded by $\log(n)^c$. This allows us to isolate $O(n/\log(n)^c)$ vectors for which further iterations are needed. A maximal basis is then computed from these vectors, based on polynomial matrix operations. The final maximal basis is obtained by appropriately merging the short and long sequences of Krylov iterates henceforth available, via fast Gaussian elimination [12, Sec. 4].

Frobenius normal form and extensions. Krylov matrices are a fundamental tool for decomposing a vector space with respect to a linear operator (see e.g. [6, 20] for detailed algorithmic treatments). The related Frobenius normal form, and the Kalman decomposition for linear dynamical systems [10, 11], are briefly discussed in Section 5. The fastest known deterministic algorithm for the Frobenius form is found in [20, Prop. 9.27]; the employed approach reveals in particular that the problem reduces to computing $O(\operatorname{loglog}(n))$ maximal Krylov bases. The cost for the Kalman decomposition is bounded by that of the computation of a constant number of Krylov bases. Thus our results improve the complexity bounds for these two problems. Matrix exponentiation is also accelerated, via its direct link with Frobenius form computation [6, Thm. 7.3].

Computational model. Throughout this paper, $\mathbb{F}$ is an effective field. The cost analyses consist in giving an asymptotic bound on the number of arithmetic operations in $\mathbb{F}$ used by the algorithm. The operations are addition, subtraction, multiplication, and inversion in the field, as well as testing whether a given field element is zero. We use that two polynomials in $\mathbb{F}[x]$ of degree bounded by d can be multiplied in $O(d \log(d) \operatorname{loglog}(d))$ operations in $\mathbb{F}$ [5, Chap. 8]. On rare occasions, mostly in Section 6, we use a multiplication time function $d \mapsto \mathsf{M}(d)$ for $\mathbb{F}[x]$ to develop somewhat more general results [5, Chap. 8]. This multiplication time function is subject to some convenient assumptions (see Section 6.1), which are satisfied in particular for an $O(d \log(d) \operatorname{loglog}(d))$ algorithm.

Notation. For an $m \times n$ matrix A, we write a_{ij} for its entry (i, j). Given sets I and J of row and column indices, $A_{I,J}$ stands for the corresponding submatrix of A; we use $*$ to denote all indices, such as in $A_{I,*}$ or $A_{*,J}$. We manipulate tuples $d \in (\mathbb{N} \cup \{-\infty\})^m$ of indices or polynomial degrees, and write $|d|$ for the sum of the entries. For a polynomial matrix $A \in \mathbb{F}[x]^{m \times n}$, the column degree $\operatorname{cdeg}(A)$ is the tuple of its column degrees $(\max_{1 \le i \le m} \deg(a_{i,j}))_{1 \le j \le n}$.

2 KERNEL BASIS, HERMITE NORMAL FORM

We recall the complexity bounds for two fundamental problems which we rely on: kernel basis and Hermite normal form. The more technical presentation of some of the other ingredients needed to manipulate polynomial matrices, such as "unbalanced" truncated inversion and multiplication, is deferred to Section 6.

2.1 Minimal kernel basis

A core tool in Algorithms 1 and 2 is the computation of minimal kernel bases of polynomial matrices [10, Sec. 6.5.4, p. 455]. The matrix fraction description in Eq. (3) can indeed be rewritten as

$$\begin{bmatrix} I - xA & -U \end{bmatrix} \begin{bmatrix} S \\ T \end{bmatrix} = 0.$$

For a matrix $F \in \mathbb{F}[x]^{n \times m}$, its (right) kernel is the $\mathbb{F}[x]$-module formed by the vectors $p \in \mathbb{F}[x]^m$ such that $Fp = 0$; it has rank $m - \operatorname{rank}(F)$. A kernel basis is said to be minimal if it is column reduced, that is, its leading matrix has full column rank [10, Sec. 6.3, p. 384]. An efficient algorithm for minimal kernel bases was described in [26], and its complexity was further analyzed in the case of a full rank input F in [9, App. B] and [16, Lem. 2.10]. We will use the following particular case of the latter result:

Lemma 2.1. ([26, Algo. 1], and analyses in [9, 16].) *Let $F \in \mathbb{F}[x]^{n \times (m+n)}$ have rank n and degree ≤ 1, with $m \in O(n)$. There is an algorithm* `MinimalKernelBasis` *which, on input F, returns a minimal kernel basis $B \in \mathbb{F}[x]^{(m+n) \times m}$ for F using $O(n^\omega)$ operations in $\mathbb{F}$. Furthermore, $|\operatorname{cdeg}(B)| \le n$.*

The complexity bound in [16] uses a multiplication time function with assumptions that are satisfied in our case (see [16, Sec. 1.1]). Apart from supporting degree > 1, the three listed references also consider the more general *shifted* reduced bases [2]. The non-shifted case is obtained by using the uniform shift $s = (1, \ldots, 1) \in \mathbb{N}^{m+n}$, for which s-reduced bases are also (non-shifted) reduced bases. This shift does satisfy the input requirement $s \ge \operatorname{cdeg}(F)$ and it has sum $|s| = m + n$; [26, Thm. 3.4] then guarantees that B has sum of s-column degrees at most $m + n$, which translates as $|\operatorname{cdeg}(B)| \le n$.

2.2 Hermite normal form

A nonsingular polynomial matrix $H \in \mathbb{F}[x]^{m \times m}$ is in (column) Hermite normal form if it is upper triangular, with monic diagonal entries, and all entries to the right of the diagonal have degree less than that of the corresponding diagonal entry: $\deg(h_{ij}) < \deg(h_{ii})$ for $1 \le i < j \le m$. Given a nonsingular matrix $T \in \mathbb{F}[x]^{m\times m}$, there is a unique matrix $H \in \mathbb{F}[x]^{m\times m}$ in Hermite normal form which can be obtained from T via unimodular column operations, meaning $TV = H$ for some $V \in \mathbb{F}[x]^{m\times m}$ with $\det(V) \in \mathbb{F} \setminus \{0\}$. It is called the Hermite normal form of T. In fact, in this paper we only seek the diagonal degrees of H (see Section 3.1.2). Since off-diagonal entries can be reduced by diagonal ones through unimodular column operations, these diagonal degrees are the same for all triangular forms unimodularly right equivalent to T.

Lemma 2.2. ([13, Prop. 3.3].) *Let T be nonsingular in $\mathbb{F}[x]^{m\times m}$. There is an algorithm* `HermiteDiagonal` *which takes T as input and returns the diagonal entries $(h_{11},\dots,h_{mm}) \in \mathbb{F}[x]^m$ of the Hermite normal form of T using $O(m^{\omega-1} n \log(n)^{c_1})$ operations in $\mathbb{F}$, where $n = \max(m, |\mathrm{cdeg}(T)|)$ and c_1 is a positive real constant.*

The algorithm as described in [13, Algo. 1] works with an equivalent lower triangular definition of the form. It involves two main tools: kernel bases [26], whose cost involves a logarithmic factor of the form $\log(n/m)^2 \operatorname{loglog}(n/m)$ [16, Lem. 2.10], and column bases [25], for which there is no analysis of the number of logarithmic factors in the literature. In Lemma 2.2 we introduce the latter as a constant $c_1 > 0$, and this remains to be thoroughly analyzed.

3 KRYLOV BASIS VIA POLYNOMIAL KERNEL

The key ingredients in our approach for computing Krylov bases transform the problem into polynomial matrix operations. In this section, we present two algorithms which involve matrix fraction descriptions as in Eq. (3) or equivalent formulations (see also Appendix B). In Section 3.1, given $A \in \mathbb{F}^{n\times n}$ and $U = [u_1 \cdots u_m]$ formed from m vectors in $\mathbb{F}^n$, Algorithm `MaxIndices` computes the maximal Krylov indices of $\mathrm{Orb}(A,U)$. It exploits the fact that these indices coincide with the degrees of certain minimal polynomial relations between the columns of $[xI - A \quad -U]$, allowing the use of polynomial matrix manipulations. In Section 3.2, we consider the problem of computing the Krylov matrix $\mathrm{K}_d(A,U)$ for a given tuple $d \in \mathbb{N}^m$. Based on Eq. (3), after computing a kernel basis to obtain S and T, Algorithm `KrylovMatrix` proceeds with a matrix series expansion, using the truncation orders given by d. Joining both algorithms, from A and U one obtains the maximal Krylov basis of $\mathrm{Orb}(A,U)$ using $O(n^\omega)$ field operations as soon as the number m of vectors forming U is in $O(n/\log(n)^c)$. This will be done explicitly in Lines 1 to 4 of Algorithm `MaxKrylovBasis` in Section 4.

3.1 Computing the maximal indices

We begin in Section 3.1.1 with a brief reminder of a useful characterization of the maximal Krylov indices in terms of linear algebra over $\mathbb{F}$. We then show how to reduce their computation to operations on polynomial matrices, by explaining that they coincide with the degrees of certain polynomials in a kernel basis (Lemma 3.1).

Linear dependencies between vectors in Krylov subspaces are translated into polynomial relations using the $\mathbb{F}[x]$-module structure of $\mathbb{F}^n$ based on $xu = Au$ for $u \in \mathbb{F}^n$ [8, Sec. 3.10]. Some of these dependencies in $\mathrm{Orb}(A,U)$ are given by the coefficients of the entries of triangular matrices $H \in \mathbb{F}[x]^{m\times m}$ such that

$$(xI - A)^{-1}U = LH^{-1} \tag{4}$$

with $L \in \mathbb{F}[x]^{n\times m}$ [10, Sec. 6.7.1, p. 476]. In particular, if L and H are right coprime then the maximal indices of $\mathrm{Orb}(A,U)$ are given by the diagonal degrees of H. This is detailed in Section 3.1.2, considering equivalently that $[L^\intercal \quad H^\intercal]^\intercal$ is a kernel basis of $[xI - A \quad -U]$. The computation of the indices follows in Section 3.1.3: it combines the kernel and Hermite form tools discussed in Section 2.

3.1.1 Keller-Gehrig's branching algorithm. The most efficient algorithm so far for computing the lexicographically largest tuple d such that $\mathrm{K}_d(A,U)$ is a Krylov basis is given in [12, Sec. 5]. For a general U (with $m \in O(n)$), its cost bound $O(n^\omega \log(n))$ is notably due to exponentiating A in order to generate Krylov iterates. Keller-Gehrig's approach uses the following characterization of the maximal indices d, which we will need (a proof can be found in Section A): for $1 \le j \le m$, d_j is the smallest integer such that

$$A^{d_j} u_j \in \mathrm{Span}(u_j, Au_j, \dots, A^{d_j-1}u_j) + \mathrm{Orb}(A, U_{*,1..j-1}). \tag{5}$$

A recursive construction allows independent and increasingly long Krylov chains $u_j, Au_j, \dots, A^{\ell-1}u_j, \ell \ge 0$ to be joined together in order to reach the maximal basis of $\mathrm{Orb}(A,U)$, with independence guaranteed by Gaussian elimination [12, Sec. 4]. We will return to this in our general Krylov basis method in Section 4.

3.1.2 Links with the Hermite normal form. The following is to be compared with known techniques for matrix fraction descriptions [10, Sec. 6.4.6, p. 424], or to a formalism occasionally used to efficiently compute matrix normal forms [23; 20, Chap. 9].

Lemma 3.1. *Given $A \in \mathbb{F}^{n\times n}$ and $U \in \mathbb{F}^{n\times m}$, let $[L^\intercal \quad H^\intercal]^\intercal$ be a kernel basis of $[xI - A \quad -U]$ such that $H \in \mathbb{F}[x]^{m\times m}$ is upper triangular. The matrix H is nonsingular and its diagonal degrees are the maximal Krylov indices of* $\mathrm{Orb}(A,U)$.

Proof. We first note that, for an upper triangular $P = [p_{i,j}]_{i,j}$ in $\mathbb{F}[x]^{m\times m}$, $(xI - A)^{-1}UP$ is a polynomial matrix if and only if

$$p_{jj}(A)u_j + \sum_{i=1}^{j-1} p_{ij}(A)u_i = 0, 1 \le j \le m. \tag{6}$$

Indeed, using expansions in $1/x$ with $(xI - A)^{-1} = \sum_{k\ge 0} A^k/x^{k+1}$, the coefficient of $1/x^{k+1}$ in the jth column of $(xI - A)^{-1}UP$ is

$$\sum_{\ell=0}^{\deg(p_{jj})} A^{\ell+k}u_j p_{jj}^{(\ell)} + \sum_{i=1}^{j-1}\sum_{\ell=0}^{\deg(p_{ij})} A^{\ell+k}u_i p_{ij}^{(\ell)}, \tag{7}$$

for any $k \ge 0$. Here $p_{ij}^{(\ell)}$ is the coefficient of degree ℓ of $p_{i,j}$, for $1 \le i \le j \le m$. The "if" direction follows: taking $k = 0$, the (expansion of the) polynomial matrix $(xI - A)^{-1}UP$ has no term in $1/x$, hence Eq. (6). Conversely, if Eq. (6) holds, multiplying it by A^k shows that the expression in Eq. (7) is zero, for any $k \ge 0$; hence the terms in $1/x^{k+1}$ are zero in the expansion of $(xI - A)^{-1}UP$, which is thus a polynomial matrix.

Now let d be the maximal indices, and let d' be the tuple of the diagonal degrees of H. Observe that H is nonsingular: since $L = (xI - A)^{-1}UH$, $Hu = 0$ for $u \ne 0$ would lead to $[L^\intercal \quad H^\intercal]^\intercal u = 0$, which is impossible by definition of bases. Thus $d' \in \mathbb{N}^m$.

To conclude, we first show that d is greater than or equal to d', and then the converse. The characterization of d given by Eq. (5) leads to linear dependencies as in Eq. (6), hence to P with diagonal degrees given by d. Since $(xI - A)^{-1}UP$ is a polynomial matrix, say R, the columns of $[R^\mathsf{T} \;\; P^\mathsf{T}]^\mathsf{T}$ are in the kernel of $[xI - A \;\; -U]$, which is generated by $[L^\mathsf{T} \;\; H^\mathsf{T}]^\mathsf{T}$, hence we must have $P = HQ$ for some polynomial matrix Q. Since both P and H are triangular we conclude that $d \geq d'$ entry-wise. On the other hand, considering H, we get dependencies as in Eq. (6) with polynomials given by the entries of H. Thanks to the fact that the d_j's reflect the shortest dependencies (see Section 3.1.1), we get that $d' \geq d$ entry-wise. ◻

3.1.3 Algorithm `MaxIndices`. Let $V \in \mathbb{F}[x]^{m\times m}$ be unimodular such that $H = TV$ is in Hermite form. The matrix $[(SV)^\mathsf{T} \;\; H^\mathsf{T}]^\mathsf{T} = [S^\mathsf{T} \;\; T^\mathsf{T}]^\mathsf{T} V$ is also a kernel basis of $[xI - A \;\; -U]$, and the correctness of the algorithm follows from Lemma 3.1 with $L = SV$.

For the complexity bound, the kernel basis at Line 1 costs $O(n^\omega)$ operations in $\mathbb{F}$ when $m \in O(n)$, according to Lemma 2.1, which also ensures $|\mathrm{cdeg}(T)| \leq n$. From Lemma 3.1, the resulting matrix T is nonsingular because its Hermite form is. Lemma 2.2 states that Line 2 can be achieved using $O(m^{\omega-1} n \log(n)^{c_1})$ operations in $\mathbb{F}$. In particular, as soon as the number m of columns of U is in $O(n/\log(n)^{c_1/(\omega-1)})$, the overall complexity bound is $O(n^\omega)$.

Algorithm 1 `MaxIndices`(A, U)

Input: $A \in \mathbb{F}^{n\times n}, U \in \mathbb{F}^{n\times m}$

Output: the tuple in $\mathbb{N}^m$ of the maximal indices of $\mathrm{Orb}(A, U)$

1: ▷ *minimal kernel basis [26, Algo. 1]* ◃
   $\left[\begin{smallmatrix} S \\ T \end{smallmatrix}\right] \leftarrow$ `MinimalKernelBasis`$([xI - A \;\; -U])$
   where $S \in \mathbb{F}[x]^{n\times m}$ and $T \in \mathbb{F}[x]^{m\times m}$

2: ▷ *diagonal entries of the Hermite normal form* ◃
   $(h_{11}, \ldots, h_{mm}) \in \mathbb{F}[x]^m \leftarrow$ `HermiteDiagonal`(T) ▷ *[13, Alg. 1]*

3: **return** $(\deg(h_{11}), \ldots, \deg(h_{mm}))$

3.2 Computing a Krylov matrix

Hereafter, for a column polynomial vector $v = \sum_j v_j x^j \in \mathbb{F}[x]^m$ and an integer $d \in \mathbb{N}$, we denote its truncation at order d by

$$v \,\mathrm{rem}\, x^d = v_0 + v_1 x + \cdots + v_{d-1} x^{d-1} \in \mathbb{F}[x]^m$$

and the corresponding matrix of coefficients by

$$\mathrm{coeffs}(v, d) = [v_0 \;\; v_1 \;\; \cdots \;\; v_{d-1}] \in \mathbb{F}^{m\times d}.$$

We extend this column-wise for $M \in \mathbb{F}[x]^{m\times k}$ and $d = (d_j)_j \in \mathbb{N}^k$, that is, $M \,\mathrm{rem}\, x^d = [M_{*,1} \,\mathrm{rem}\, x^{d_1} \;\; \cdots \;\; M_{*,k} \,\mathrm{rem}\, x^{d_k}]$.

Algorithm `KrylovMatrix` uses polynomial matrix subroutines at Lines 2 and 3. They are handled in detail in Sections 6.2 and 6.3.

Lemma 3.2. *Algorithm* `KrylovMatrix` *is correct. If m and $|d|$ are both in $O(n)$, it uses $O(n^\omega + m^{\omega-2} n^2 \log(n)^4)$ operations in $\mathbb{F}$.*

Proof. Knowing by Lemma 3.3 that $T(0)$ is invertible (hence T is nonsingular), the correctness follows from the series expansion

$$ST^{-1} = (I - xA)^{-1} U = \sum_{k\geq 0} x^k A^k U$$

mentioned in Eq. (3); here the first equality comes from the kernel relation $[I - xA \;\; -U][S^\mathsf{T} \;\; T^\mathsf{T}]^\mathsf{T} = 0$, by construction in Line 1. Since $T(0)$ is invertible, Lines 2 and 3 compute $P = SQ \,\mathrm{rem}\, x^d = ST^{-1} \,\mathrm{rem}\, x^d$, according to Propositions 6.3 and 6.4. The above equation yields $\mathrm{coeffs}(P_{*,j}, d_j) = [u_j \;\; Au_j \;\; \cdots \;\; A^{d_j-1}u_j]$, hence the correctness of the output formed at Line 4.

We turn to the complexity analysis. We assume $m \leq |d|$ without loss of generality: the columns of U corresponding to indices j with $d_j = 0$ could simply be ignored in Algorithm `KrylovMatrix`, reducing to a case where all entries of d are strictly positive.

By Lemma 2.1, Line 1 uses $O(n^\omega)$ operations in $\mathbb{F}$ and ensures $|\mathrm{cdeg}(K)| \leq n$. In particular, $|\mathrm{cdeg}(S)| \leq n$ and $|\mathrm{cdeg}(T)| \leq n$.

The latter bound ensures that the generic determinantal degree $\Delta(T)$ is in $O(n)$ (see Lemma 6.1), hence we can use the second cost bound in Proposition 6.3: Line 2 computes $Q = T^{-1} \,\mathrm{rem}\, x^d$ using

$$O\left(m^\omega \,\mathsf{M}\!\left(\frac{n}{m}\right)\left(\log(n) + \left\lceil\frac{|d|}{n}\right\rceil \log(m)\log(|d|)\right)\right) \tag{8}$$

field operations. The second cost bound in Proposition 6.4 states that Line 3 computes $SQ \,\mathrm{rem}\, x^d$ using $O(m^{\omega-2} n \,\mathsf{M}(n + |d|))$ operations in $\mathbb{F}$. Summing the latter bound with that in Eq. (8) yields a cost bound for Algorithm `KrylovMatrix` when the only assumption on d is $m \leq |d|$. Now, assuming further $|d| \in O(n)$, the latter bound becomes $O(m^{\omega-2} n \,\mathsf{M}(n))$, whereas the one in Eq. (8) simplifies as $O(m^\omega \,\mathsf{M}(\frac{n}{m}) \log(n) \log(m))$. The claimed cost bound then follows from $\mathsf{M}(n) \in O(n \log(n) \,\mathrm{loglog}(n))$. ◻

Algorithm 2 `KrylovMatrix`(A, U, d)

Input: $A \in \mathbb{F}^{n\times n}, U \in \mathbb{F}^{n\times m}$, $d = (d_1, \ldots, d_m) \in \mathbb{N}^m$

Output: the Krylov matrix $\mathrm{K}_d(A, U) \in \mathbb{F}^{n\times(d_1+\cdots+d_m)}$

1: ▷ *minimal kernel basis [26, Algo. 1]* ◃
   $\left[\begin{smallmatrix} S \\ T \end{smallmatrix}\right] \leftarrow$ `MinimalKernelBasis`$([I - xA \;\; -U])$
   where $S \in \mathbb{F}[x]^{n\times m}$ and $T \in \mathbb{F}[x]^{m\times m}$

2: ▷ *column-truncated inverse T^{-1} rem x^d, detailed in Section 6.2* ◃
   $Q \in \mathbb{F}[x]^{m\times m} \leftarrow$ `TruncatedInverse`(T, d)

3: ▷ *column-truncated product SQ rem x^d, detailed in Section 6.3* ◃
   $P \in \mathbb{F}[x]^{n\times m} \leftarrow$ `TruncatedProduct`(S, Q, d)

4: ▷ *linearize columns of P into a constant matrix and return* ◃
   return $[\mathrm{coeffs}(P_{*,1}, d_1) \mid \cdots \mid \mathrm{coeffs}(P_{*,m}, d_m)] \in \mathbb{F}^{n\times|d|}$

Lemma 3.3. *For T as in Line 1 of Algorithm 2, $T(0)$ is invertible.*

Proof. Let $B = [S^\mathsf{T} \;\; T^\mathsf{T}]^\mathsf{T}$. As a kernel basis, B can be completed into a basis of $\mathbb{F}[x]^{m+n}$ [14, Lem. 7.4, p148]: there is $C \in \mathbb{F}[x]^{(m+n)\times n}$ such that $\det([B \;\; C]) = \det([B(0) \;\; C(0)]) = 1$. We conclude by deducing that any vector $v \in \mathbb{F}^m$ such that $T(0)v = 0$ must be zero: from $[I \;\; -U]B(0) = 0$ we get $S(0)v = 0$, hence $B(0)v = 0$, which implies $v = 0$ since $[B(0) \;\; C(0)]$ is invertible. ◻

4 PREPROCESSING OF SMALL INDICES

In this section we consider the case where the number m of vectors to be iterated can be large. Say for instance $m = \Theta(n)$: then, terms with logarithmic factors in the costs of Algorithms 1 and 2, such as $O(m^{\omega-1} n \log(n)^{c_1})$ or $O(m^{\omega-2} n^2 \log(n)^4)$, are not negligible anymore compared to the cost $O(n^\omega)$ of the kernel basis computations.

Algorithm `MaxKrylovBasis` computes the maximal Krylov basis of $\mathrm{Orb}(A, U)$. It involves a preprocessing phase based on the Keller-Gehrig branching algorithm [12, Thm 5.1], which is terminated after a small number ℓ of iterations. This ensures that after this phase, no more than $m = n/2^\ell$ vectors need to be iterated further, which can then be performed by Algorithms 1 and 2. Specifically, setting $\ell = \lceil c \operatorname{loglog}(n) \rceil$ with $c = \max(4/(\omega-2), c_1/(\omega-1))$ yields $m \approx n/\log(n)^c$, ensuring that Algorithms 1 and 2 run in $O(n^\omega)$.

If m is sufficiently small from the start, the preprocessing phase is skipped: a direct call to the latter algorithms is made. Otherwise, a **for** loop (Line 10 of Algorithm `MaxKrylovBasis`) performs ℓ loop iterations akin to those used in [12, Sec. 5]. At the end of this loop, the indices J of the vectors requiring further iterations are identified (Line 23), and processed via Algorithms 1 and 2 (Line 24 and 25). Finally, both temporary sequences of iterates, the "long" ones (indices in J) and "short" ones (indices not in J), are merged via Gaussian elimination and the maximal basis is obtained.

Algorithm 3 `MaxKrylovBasis`(A, U)

Input: $A \in \mathbb{F}^{n\times n}$, $U = [u_1 \ \cdots \ u_m] \in \mathbb{F}^{n\times m}$
Output: the maximal Krylov basis of $\operatorname{Orb}(A, U)$

1: $c \leftarrow \max(4/(\omega-2), c_1/(\omega-1))$; $t \leftarrow \log_2(n)^c$
2: **if** $m \le n/t$ **then**
3: $\quad d \leftarrow$ `MaxIndices`(A, U)
4: $\quad K \leftarrow$ `KrylovMatrix`(A, U, d); **return** K
5: **end if**
6: ▷ *Preprocessing phase in a Keller-Gehrig fashion [12]* ◁
7: $\ell \leftarrow \lceil \log_2(t) \rceil$
8: $V^{(0)} \leftarrow U$; $\delta \leftarrow (1, \ldots, 1) \in \mathbb{Z}^m$
9: $B \leftarrow A$
10: **for** $i \leftarrow 0, \ldots, \ell - 1$ **do**
11: $\quad$ Write $V^{(i)} = [V_1^{(i)} \ V_2^{(i)} \ \cdots \ V_m^{(i)}]$ where $V_j^{(i)} = \mathrm{K}_{\delta_j}(A, U_{*,j}) \in \mathbb{F}^{n\times \delta_j}$
12: $\quad J = \{j_1 < \cdots < j_s\} \leftarrow \{j \in \{1, \ldots, m\} \mid \delta_j = 2^i\}$
13: $\quad [W_{j_1}^{(i)} \ \cdots \ W_{j_s}^{(i)}] \leftarrow B[V_{j_1}^{(i)} \ \cdots \ V_{j_s}^{(i)}]$
14: $\quad W_j^{(i)} \leftarrow [\,] \in \mathbb{F}^{n\times 0}$ for $j \notin J$
15: $\quad Z \leftarrow [V_1^{(i)} \ W_1^{(i)} \ \cdots \ V_m^{(i)} \ W_m^{(i)}]$
16: $\quad C \leftarrow \operatorname{ColRankProfile}(Z)$
17: $\quad \delta \leftarrow (\delta_1, \ldots, \delta_m)$ such that δ_j is maximal with $\mathrm{K}_{\delta_j}(A, u_j) = Z_{*, C\cap\{b..b+\delta_j-1\}}$ for some b
18: $\quad V^{(i+1)} \leftarrow [V_1^{(i+1)} \ \cdots \ V_m^{(i+1)}]$ where each $V_j^{(i+1)} \leftarrow \mathrm{K}_{\delta_j}(A, u_j)$ is copied from Z
19: $\quad B \leftarrow B^2$
20: **end for**
21: ▷ *Here,* $V^{(\ell)} = \mathrm{K}_\delta(A, U)$ *and* $\delta = (\delta_1, \ldots, \delta_m) \in \{0, \ldots, 2^\ell\}^m$ *is lexicographically maximal such that* $\mathrm{K}_\delta(A, U)$ *has full rank* ◁
22: ▷ *Further iterations for selected vectors, using polynomial matrices* ◁
23: $J = \{j_1 < \cdots < j_s\} \leftarrow \{j \in \{1, \ldots, m\}, \delta_j = 2^\ell\}$
24: $d \leftarrow$ `MaxIndices`$(A, U_{*,J})$
25: $[K_{j_1} \ \cdots \ K_{j_s}] \leftarrow$ `KrylovMatrix`$(A, U_{*,J}, d)$
26: ▷ *Final merge* ◁
27: $K_j \leftarrow V_j^{(\ell)}$ for $j \notin J$; $\ K \leftarrow [K_1 \ \cdots \ K_m]$
28: **return** $K_{*,\operatorname{ColRankProfile}(K)}$

Theorem 4.1. *Algorithm* `MaxKrylovBasis` *is correct. If* $m = O(n)$*, it uses* $O(n^\omega \operatorname{loglog}(n))$ *operations in* $\mathbb{F}$*, and if* $m = O(n/\log(n)^c)$ *for the constant* $c > 0$ *in Line 1, it uses* $O(n^\omega)$ *operations in* $\mathbb{F}$.

Proof. As in Keller-Gehrig's branching algorithm [12], the loop invariant is $V^{(i)} = \mathrm{K}_\delta(A, U)$ with $\delta \in \{0, \ldots, 2^i\}^m$ lexicographically maximal such that $V^{(i)}$ has full rank. By induction, it remains valid upon exiting the loop, as stated in Line 21. At this point, $s = \#J$ is the number of vectors left to iterate. Indeed, $j \notin J$ means that a linear relation of the type

$$A^{\delta_j} u_j \in \operatorname{Span}(u_j, Au_j, \ldots, A^{\delta_j - 1} u_j) + \operatorname{Orb}(A, U_{*,1..j-1}) \qquad (9)$$

has already been found (recall Eq. (5)).

A maximal Krylov basis for the subset of vectors given by J is then computed using the algorithms of Section 3. All the Krylov iterates that form the matrix K at Line 27 are considered at orders higher than or equal to the final maximal ones, because relations of the type Eq. (9) have been detected for all j. Thus, a column rank profile computation yields the maximal indices and selects the vectors for the maximal basis.

The choice for the parameter t ensures that the for-loop is executed using $O(n^\omega \operatorname{loglog}(n))$ field operations. On the other hand, Algorithms 1 and 2 are called with $m = s \le n/2^\ell \le n/\log(n)^c$, and therefore run in $O(n^\omega)$ field operations. □

In addition, Algorithm `MaxKrylovBasis` can be adapted so as to compute a Krylov matrix $\mathrm{K}_d(A, U)$ for given orders d (not necessarily the maximal indices) that are provided as additional input.

Corollary 4.2. *Given* $A \in \mathbb{F}^{n\times n}$, $U \in \mathbb{F}^{n\times m}$*, and* $d \in \mathbb{N}^m$*, with* m *and* $|d|$ *both in* $O(n)$*, the Krylov matrix* $\mathrm{K}_d(A, U)$ *can be computed using* $O(n^\omega \operatorname{loglog}(n))$ *operations in* $\mathbb{F}$*, or* $O(n^\omega)$ *operations in* $\mathbb{F}$ *if* $m \in O(n/\log(n)^c)$ *where* c *is a positive real constant.*

This only requires the following modifications:

(1) Line 16 and 17 should be replaced by
for all $j \in J$ **do** $\delta_j \leftarrow \min(2\delta_j, d_j)$
(2) Line 28 should be replaced by
return K
(3) remove Lines 3 and 24.

5 FROBENIUS AND KALMAN FORMS

In this section we discuss some consequences of our new algorithms for Krylov bases, namely some improved complexity bounds for directly related problems.

5.1 Frobenius normal form

Generically, the Frobenius normal form of an $n \times n$ matrix can be computed using $O(n^\omega)$ operations in $\mathbb{F}$ [12, Sec. 6; 18], and the approach in [18] mainly provides a Las Vegas probabilistic algorithm in $O(n^\omega)$. It is still an open question to obtain the same complexity bound with a deterministic algorithm, and also to compute an associated transformation matrix. Our results allow us to make some progress on both aspects.

Since this is often a basic operation for these problems, we can already note that from Lemma 3.1 and its proof, the minimal polynomial of a vector $u \in \mathbb{F}^n$ can be computed in $O(n^\omega)$, via Lemma 2.1. Indeed this minimal polynomial is the last entry, made monic, of any kernel basis (in this case a single vector) of $[xI - A \quad -u]$.

Our algorithms make it possible to obtain $O(n^\omega)$ for the Frobenius form with associated transformation matrix in a special case.

The general cost bound $O(n^\omega \operatorname{loglog}(n))$ is achieved in [19, Theorem 7.1] with a probabilistic algorithm. To have a transformation, we can first compute the Frobenius form alone using $O(n^\omega)$ operations. If it has m non-trivial blocks and $U \in \mathbb{F}^{n\times m}$ is chosen uniformly at random, then we know that a transformation matrix can be computed from the maximal Krylov basis of $\operatorname{Orb}(A, U)$ using $O(n^\omega)$ operations [6, Thm. 2.5 & 4.3]. So we have the following.

Corollary 5.1. *Let $A \in \mathbb{F}^{n\times n}$ with $\#\mathbb{F} \geq n^2$, and assume that its Frobenius normal form has $m \in O(n/\log(n)^c)$ non-trivial blocks. A transformation matrix to the form can be computed by a Las Vegas probabilistic algorithm using $O(n^\omega)$ operations in $\mathbb{F}$.*

The fastest deterministic algorithm to compute a transformation to Frobenius form is given in [21] (see also [20, Chap. 9]), with a cost of $O(n^\omega \log(n) \operatorname{loglog}(n))$. This cost is essentially $O(\operatorname{loglog}(n))$ computations of maximal Krylov bases, plus $O(n^\omega)$ operations.

Corollary 5.2. *Given $A \in \mathbb{F}^{n\times n}$, a transformation matrix to the Frobenius normal form of A can be computed using $O(n^\omega \operatorname{loglog}(n)^2)$ operations in $\mathbb{F}$.*

Finally, once the Frobenius form and a transformation matrix are known, and given an integer $k \geq 0$, computing A^k costs $O(n^\omega)$ plus $O(\log(k)\,\mathsf{M}(n))$ field operations (see e.g. [6, Cor. 7.4]). As mentioned in Section 1, here $d \mapsto \mathsf{M}(d)$ is a multiplication time function for $\mathbb{F}[x]$. So, in total, computing A^k costs $O(n^\omega \operatorname{loglog}(n)^2)$ if $\log(k) \in O(n^{\omega-1-\varepsilon})$. The evaluation of a polynomial $p \in \mathbb{F}[x]$ at A can also be considered in a similar way, using, for example, the analysis of [6, Thm. 7.3].

5.2 Kalman decomposition

The study of the structure of linear dynamical systems in control theory is directly related to Krylov spaces and matrix polynomial forms [10]. For example, the link we use between maximal indices and Hermite form degrees originates from this correspondence.

Our work could be continued to show that the complexity bound $O(n^\omega \operatorname{loglog}(n))$ in Theorem 4.1 could be applied to the computation of a Kalman decomposition [11; 10, Sec. 2.4.2, p. 128]. This is beyond the scope of this paper, so we will not go into detail about it here. However, we can specify the main ingredient. Given A and U with $\dim(\operatorname{Orb}(A, U)) = \nu$, we want to transform the system (A, U) according to [10, Sec. 2.4.2, Eq. (11), p. 130]:

$$P^{-1}AP = \begin{bmatrix} A_c & A_1 \\ 0 & A_2 \end{bmatrix},\ P^{-1}U = \begin{bmatrix} U_c \\ 0 \end{bmatrix},$$

where A_c is $\nu \times \nu$, U_c is $\nu \times m$, and P is invertible in $\mathbb{F}^{n\times n}$. The matrix P can be formed by a Krylov basis of $\operatorname{Orb}(A, U)$ and a matrix with $n - \nu$ independent columns not in $\operatorname{Orb}(A, U)$. The general decomposition is obtained by combining a constant number of such transformations and basic matrix operations to decompose (A, U).

6 POLYNOMIAL MATRIX SUBROUTINES

We now detail the subroutines used in Algorithm `KrylovMatrix`. Section 6.1 introduces some convenient notation for complexity bounds. Section 6.2 combines high-order lifting [22] and partial linearization [7] to compute truncated inverse expansions, while Section 6.3 focuses on truncated matrix products. In both cases, the difficulty towards efficiency lies in the presence of unbalanced degrees and unbalanced truncation orders.

6.1 Complexity helper functions

We briefly recall notation and assumptions about cost functions; for more details, we refer to [5, Chap. 8] for the general framework, and to [22, Sec. 2] and [16, Sec. 1.1] for polynomial matrices specifically.

In what follows, we assume fixed multiplication algorithms:

- for polynomials in $\mathbb{F}[x]$, with cost function $\mathsf{M}(d)$ when the input polynomials have degree at most d;
- for matrices in $\mathbb{F}^{m\times m}$, with cost $O(m^\omega)$;
- for polynomial matrices in $\mathbb{F}[x]^{m\times m}$, with cost function $\mathsf{MM}(m, d)$ when the input matrices have degree at most d.

To simplify our analyses and the resulting bounds, we make the same assumptions as in the above references. In particular, $\omega > 2$, $\mathsf{M}(\cdot)$ is superlinear, and $\mathsf{MM}(m, d) \in O(m^\omega\,\mathsf{M}(d))$.

We will also use the function $\overline{\mathsf{MM}}(m, d)$ from [22, Sec. 2]; as noted in this reference, the above-mentioned assumptions imply $\overline{\mathsf{MM}}(m, d) \in O(m^\omega\,\mathsf{M}(d)\log(d))$.

6.2 Polynomial matrix truncated inverse

In Algorithm `KrylovMatrix`, Line 2 asks to find terms of the power series expansion of P^{-1}, for an $m\times m$ polynomial matrix P with $P(0)$ invertible. Customary algorithms for this task, depending on the range of parameters (m, $\deg(P)$, truncation order), include a matrix extension of Newton iteration [15; 5, Chap. 9], or matrix inversion [27] followed by Newton iteration on the individual entries.

Here, a first obstacle towards efficiency comes from the heterogeneity of truncation orders: one seeks the first d_j terms of the jth column of the expansion of P^{-1}, for some prescribed $d = (d_j)_j \in \mathbb{N}^m$ which may have unbalanced entries. In the extreme case $d = (d_1, 0, \ldots, 0)$, the task becomes the computation of many initial terms of the expansion of $P^{-1}[1\ 0\ \cdots\ 0]^\mathsf{T}$, the first column of P^{-1}. This is handled efficiently via high-order lifting techniques [22, Sec. 9]. Our solution for a general tuple d is to rely on cases where the high-order lifting approach is efficient, by splitting the truncation orders into subsets of the type $\{j \in \{1, \ldots, m\} \mid 2^{k-1}|d|/m < d_j \leq 2^k|d|/m\}$, for only logarithmically many values of k. Observe that this subset has cardinality less than $m/2^{k-1}$: higher truncation orders involve fewer columns of the inverse.

A second obstacle is due to the heterogeneity of the degrees in the matrix P itself. In the context of Algorithm `KrylovMatrix`, P may have unbalanced column degrees, but they are controlled to some extent: their sum is at most n (the dimension of the matrix A). Whereas such cases were not handled in the original description of high-order lifting, the partial linearization tools described in [7, Sec. 6] allow one to deal with this obstacle. For example, this was applied in [17, Lem. 3.3], yet in a way that is not efficient enough for the matrices P encountered here: this reference targets low average row degree for P, whereas here our main control is on the average column degree. Here, following this combination of [22, Sec. 9] and [7, Sec. 6], we present an algorithm which supports a more general unbalancedness of degrees of P. In the next two lemmas, we summarize the properties we will use from the latter references.

Lemma 6.1 ([7, Sec. 6]). *Let $P \in \mathbb{F}[x]^{m\times m}$ be nonsingular. Consider its so-called generic determinantal degree $\Delta(P)$,*

$$\Delta(P) = \max_{\pi\in\mathfrak{S}_m} \sum_{\substack{1\le i\le m\\ A_{i,\pi_i}\neq 0}} \deg(A_{i,\pi_i}) \le |\mathrm{cdeg}(P)|.$$

One can build, without using field operations, a matrix $\bar{P} \in \mathbb{F}[x]^{\bar{m}\times\bar{m}}$ of degree $\le \lceil \Delta(P)/m \rceil$ and size $m \le \bar{m} < 3m$, which is such that $\det(P) = \det(\bar{P})$ and P^{-1} is the principal $m \times m$ submatrix of $\bar{P}^{-1}$.

Lemma 6.2 ([22, Sec. 9]). *Algorithm* `SeriesSol` *[22, Alg. 4] takes as input $P \in \mathbb{F}[x]^{m\times m}$ of degree t with $P(0)$ invertible, $V \in \mathbb{F}[x]^{m\times n}$, and $s \in \mathbb{Z}_{>0}$, and returns the expansion $(P^{-1}V) \operatorname{rem} x^{st}$ using*

$$O\left(\log(s+1)\left\lceil \frac{sn}{m}\right\rceil \mathsf{MM}(m,t) + \overline{\mathsf{MM}}(m,t)\right)$$

operations in $\mathbb{F}$. The term $\overline{\mathsf{MM}}(m,t)$ comes from a call to [22, Alg. 1] which is independent of V, namely `HighOrderComp`$(P, \lceil \log_2(s)\rceil - 1)$.

Proof. There is no operation needed when $t = 0$, i.e., for a constant matrix P. For $t > 0$, this follows from [22, Prop. 15] and the paragraph that precedes it, applied with $X = x^t$. Indeed this is a valid choice since X has degree $\deg(P)$ and is coprime with $\det(P)$. The corresponding algorithm in [22] has two requirements on s, which are easily lifted: it should be a power of 2 (one can compute with the next power of 2, i.e. $2^{\lceil \log_2(s)\rceil}$, and then truncate at the desired order) and it should be at least 4 (the case $s \in O(1)$ is handled in $O(\lceil n/m\rceil\, \mathsf{MM}(m,t))$ via a direct Newton iteration). □

Proposition 6.3. *Let $P \in \mathbb{F}[x]^{m\times m}$ with $P(0)$ invertible, and let $d = (d_j)_j \in \mathbb{N}^m$. Let t be the degree of the partially linearized matrix $\bar{P}$ as in Lemma 6.1, thus with $t \le \lceil \Delta(P)/m\rceil$. Algorithm* `TruncatedInverse` *uses $O(m^\omega)$ operations in $\mathbb{F}$ if $\deg(P) = 0$, and*

$$O\left(\overline{\mathsf{MM}}(m,t) + \left\lceil \frac{|d|}{mt}\right\rceil \mathsf{MM}(m,t)\log(m)\log\left(m + \frac{|d|}{t}\right)\right)$$

operations in $\mathbb{F}$ if $\deg(P) > 0$ (which implies $t > 0$). It returns $P^{-1} \operatorname{rem} x^d$, the power series expansion of P^{-1} with column j truncated at order d_j. If n is a parameter such that m and $\Delta(P)$ are both in $O(n)$, the above bound is in

$$O\left(m^\omega\, \mathsf{M}\!\left(\frac{n}{m}\right)\left(\log(n) + \left\lceil\frac{|d|}{n}\right\rceil \log(m)\log(m+|d|)\right)\right).$$

Proof. When $d = (0,\dots,0)$, the algorithm performs no field operations and returns the zero matrix (see Line 2). When $d \neq 0$ and $\deg(P) = 0$ (hence $t = 0$), Line 3 correctly computes $P^{-1} \operatorname{rem} x^d$ in complexity $O(m^\omega)$, which is within the claimed cost since $m^\omega \in O(\mathsf{MM}(m,0))$. From here on, assume $d \neq 0$ and $\deg(P) > 0$.

The sets $J_1,\dots,J_m$ built at Line 5 are disjoint and, since $\max_j d_j \le |d| \le 2^\ell\delta$, they are such that $J_1 \cup \cdots \cup J_m = \{1,\dots,m\}$. Note also that the cardinality $n_k = \#J_k$ is less than $m/2^{k-1}$. We claim that at the end of the kth iteration of the main loop, $Q_{*,j}$ is the column j of the sought output $P^{-1} \operatorname{rem} x^d$ for all $j \in J_1 \cup \cdots \cup J_k$, which implies the correctness of the algorithm. This claim follows from Lemma 6.1. Indeed, writing $t = \deg(\bar{P})$, since the principal $m\times m$ submatrix of $\bar{P}^{-1}$ is P^{-1}, Line 19 computes, in the top m rows of F, all columns $j \in J_k$ of P^{-1} truncated at order $\lceil 2^k\delta/t\rceil t$, which is at least the target order d_j. The subsequent Line 20 further truncates to shave off the possible extraneous expansion terms, and also selects the relevant rows of F. Note that $t > 0$: if $\bar{P}$ was constant, then the principal $m \times m$ submatrix of $\bar{P}^{-1}$ would be constant, i.e. P^{-1} would be constant, which is not the case since $\deg(P) > 0$.

Algorithm 4 `TruncatedInverse`(P, d)

Input: $P \in \mathbb{F}[x]^{m\times m}$ with $P(0)$ invertible, $d = (d_1,\dots,d_m) \in \mathbb{N}^m$
Output: the column-truncated inverse $P^{-1} \operatorname{rem} x^d \in \mathbb{F}[x]^{m\times m}$

1: $Q \leftarrow$ zero matrix in $\mathbb{F}[x]^{m\times m}$ ▷ *stores the result*
2: **if** $d = (0,\dots,0)$ **then return** Q
3: **if** $\deg(P) = 0$ **then** ▷ *constant matrix inversion, $O(m^\omega)$*
4: $\quad Q \leftarrow P^{-1}$; $Q_{*,j} \leftarrow 0$ for all j with $d_j = 0$; **return** Q
5: ▷ *build partition of $\{1,\dots,m\}$ based on truncation order* ◁
6: $\delta \leftarrow |d|/m$; $\ell \leftarrow \lceil \log_2(m)\rceil$
7: $J_1 \leftarrow \{j \in \{1,\dots,m\} \mid d_j \le 2\delta\}$; $n_1 \leftarrow \#J_1$
8: **for** $k \leftarrow 2,\dots,\ell$ **do**
9: $\quad J_k \leftarrow \{j \in \{1,\dots,m\} \mid 2^{k-1}\delta < d_j \le 2^k\delta\}$; $n_k \leftarrow \#J_k$
10: **end for**
11: ▷ *partial linearization, note that $t = \deg(\bar{P}) > 0$* ◁
12: $\bar{P} \in \mathbb{F}[x]^{\bar{m}\times\bar{m}} \leftarrow$ matrix built from P as in Lemma 6.1
13: $t \leftarrow \deg(\bar{P})$; **for** $k = 1,\dots,\ell$ **do** $s_k \leftarrow \lceil 2^k\delta/t\rceil$
14: ▷ *store high-order components to avoid redundant iterations* ◁
15: $C \leftarrow$ `HighOrderComp`$[x^t](\bar{P}, \lceil\log_2(s_\ell)\rceil - 1)$ ▷ *[22, Alg. 1]*
16: ▷ *main loop: iteration k handles $Q_{*,j}$ for j in J_k* ◁
17: **for** $k \leftarrow 1, 2,\dots,\ell$ **do**
18: $\quad E \in \mathbb{F}^{\bar{m}\times n_k} \leftarrow (I_{\bar{m}})_{*,J_k}$ ▷ *columns of $\bar{m}\times\bar{m}$ identity matrix*
19: $\quad F \in \mathbb{F}[x]^{\bar{m}\times n_k} \leftarrow$ `SeriesSol`$(\bar{P}, E, s_k)$, using the pre-computed high-order components C ▷ *Lemma 6.2*
20: $\quad Q_{*,J_k} \leftarrow$ first m rows of F, all truncated at order $(d_j)_{j\in J_k}$
21: **end for**
22: **return** Q

As noted in Lemma 6.2, the kth call to `SeriesSol` involves a call to `HighOrderComp`, which does not depend on the matrix E at this iteration and which will re-compute the same high order components as the previous iterations, plus possibly one new such component. To avoid this redundancy, we pre-compute all required high-order components before the main loop at Line 15.

As for complexity, only Lines 15 and 19 use arithmetic operations. The construction of $\bar{P}$ in Lemma 6.1 implies $\det(\bar{P}(0)) = \det(\bar{P})(0) = \det(P)(0) \neq 0$, hence we can apply Lemma 6.2. Here, $\bar{P}$ is $\bar{m}\times\bar{m}$ with $m \le \bar{m} < 3m$, and $t = \deg(\bar{P}) \le \lceil \Delta(P)/m\rceil$.

Hence, using notation $s_k = \lceil 2^k\delta/t\rceil$, the complexity is within

$$O\left(\overline{\mathsf{MM}}(m,t) + \sum_{k=1}^{\ell} \log(s_k+1)\left\lceil\frac{s_k n_k}{m}\right\rceil \mathsf{MM}(m,t)\right), \text{ with}$$

$$\sum_{k=1}^{\ell}\log(s_k+1)\left\lceil\frac{s_k n_k}{m}\right\rceil \in O\left(\lceil\delta/t\rceil\left(\ell^2 + \ell\log(\lceil\delta/t\rceil)\right)\right)$$

thanks to the upper bounds $\log_2(s_k+1) \le \ell + 1 + \log_2(\lceil\delta/t\rceil)$ and $\lceil s_k n_k/m\rceil \le \lceil s_k/2^{k-1}\rceil = \lceil 2\delta/t\rceil$.

To obtain the claimed general cost bound, it remains to note that

$$\ell^2 + \ell\log(\lceil\delta/t\rceil) \in O\left(\log(m)\log(m\lceil\delta/t\rceil)\right),$$

with $m\lceil\delta/t\rceil \in O(m + |d|/t)$. For the simplified bound, we first use $t \ge 1$ to bound $\log(m + |d|/t)$ by $\log(m + |d|)$. The assumptions on the introduced parameter n allow us to write $t \in O(n/m)$. In particular, $\overline{\mathsf{MM}}(m,t)$ is in $O(m^\omega\,\mathsf{M}(n/m)\log(n/m))$, which is within the claimed bound. It remains to observe that

$$\begin{aligned}\left\lceil\frac{|d|}{mt}\right\rceil \mathsf{MM}(m,t) &\in O\left(\left(1+\frac{|d|}{mt}\right)m^\omega\,\mathsf{M}(t)\right)\\ &\subseteq O\left(m^\omega\,\mathsf{M}\!\left(\frac{n}{m}\right)+\frac{|d|}{m}m^\omega\frac{\mathsf{M}(t)}{t}\right)\\ &\subseteq O\left(m^\omega\,\mathsf{M}\!\left(\frac{n}{m}\right)+\frac{|d|}{n}m^\omega\,\mathsf{M}\!\left(\frac{n}{m}\right)\right)\subseteq O\left(\left\lceil\frac{|d|}{n}\right\rceil m^\omega\,\mathsf{M}\!\left(\frac{n}{m}\right)\right).\end{aligned}$$

Here we have used the superlinearity assumption on $\mathsf{M}(\cdot)$, which gives us $\frac{\mathsf{M}(t)}{t}\in O(\frac{\mathsf{M}(n/m)}{n/m})$. ☐

6.3 Polynomial matrix truncated product

Proposition 6.4. *Given $F\in\mathbb{F}[x]^{n\times m}$, $G\in\mathbb{F}[x]^{m\times m}$, and $d=(d_j)_j\in\mathbb{N}^m$, Algorithm* `TruncatedProduct` *uses*

$$O\left(\sum_{0\le k<\lceil\log_2(m)\rceil}\left\lceil 2^k\frac{n}{m}\right\rceil \mathsf{MM}\left(2^{-k}m, 2^k\left\lceil\frac{D}{m}\right\rceil\right)\right)$$

operations in $\mathbb{F}$ and returns the truncated product $(FG)\operatorname{rem} x^d$ that is, FG with column j truncated at order d_j. Here, D is the maximum between $|d|$ and the sum of the degrees of the nonzero columns of F. If m is both in $O(n)$ and $O(D)$, this cost bound is in $O(m^{\omega-2}n\,\mathsf{M}(D))$.

Proof. For convenience, we denote by I_k and J_k the sets I and J defined at the iteration k of the main loop of the algorithm. Let also $R^{(k)}$ be the matrix R at the beginning of iteration k, and $R^{(\ell)}$ be the output R. Let $F^{(0:k)}=F\operatorname{rem} x^{2^k\delta}$ for $1\le k\le\ell$, with in particular $F^{(0:\ell)}=F$ since $\max_j d_j\le|d|\le m\delta\le 2^\ell\delta$.

At the beginning of the first iteration, $R^{(1)}=(F^{(0)}G)\operatorname{rem} x^d=(F^{(0:1)}G)\operatorname{rem} x^d$. Then, to prove the correctness of the algorithm, we let $k\in\{1,\ldots,\ell\}$ and assume $R^{(k)}=(F^{(0:k)}G)\operatorname{rem} x^d$, and we show $R^{(k+1)}=(F^{(0:k+1)}G)\operatorname{rem} x^d$.

First, consider $j\notin J_k$. The column j of $R^{(k)}$ is not modified by iteration k, i.e. $R^{(k)}_{*,j}=R^{(k+1)}_{*,j}$. On the other hand, one has $d_j<2^k\delta$, hence $F^{(0:k)}\operatorname{rem} x^{d_j}=F^{(0:k+1)}\operatorname{rem} x^{d_j}$. We obtain

$$R^{(k+1)}_{*,j}=R^{(k)}_{*,j}=(F^{(0:k)}G_{*,j})\operatorname{rem} x^{d_j}=(F^{(0:k+1)}G_{*,j})\operatorname{rem} x^{d_j}.$$

meaning that the sought equality holds for the columns $j\notin J_k$.

Now, consider $j\in J_k$. For $i\notin I_k$, one has $\operatorname{cdeg}(F_{*,i})<2^k\delta$, hence $F^{(k)}_{*,i}=0$: it follows that $F^{(k)}G_{*,j}=F^{(k)}_{*,I}G_{I,j}$. Thus Line 9 computes

$$\begin{aligned}R^{(k+1)}_{*,j}&\leftarrow R^{(k)}_{*,j}+\left((F^{(k)}_{*,I}G_{I,j})\operatorname{rem} x^{d_j-2^k\delta}\right)x^{2^k\delta}\\ &=(F^{(0:k)}G_{*,j})\operatorname{rem} x^{d_j}+(x^{2^k\delta}F^{(k)}G_{*,j})\operatorname{rem} x^{d_j}\\ &=(F^{(0:k+1)}G_{*,j})\operatorname{rem} x^{d_j}.\end{aligned}$$

This completes the proof of correctness.

For complexity, note that the definition of D gives $\#I_k\le 2^{-k}m$ and $\#J_k\le 2^{-k}m$; also, only Lines 4 and 9 use arithmetic operations.

At Line 4, we first compute $F^{(0)}G$, then truncate. The left and right matrices in this product are respectively $n\times m$ of degree $<2\delta$, and $m\times m$ with sum of column degrees $\le m\delta$. The product can be performed by expanding the columns of G into $\le 2m$ columns all of degree $\le\delta$, which leads to the complexity $O(\lceil\frac{n}{m}\rceil\,\mathsf{MM}(m,\delta))$.

Algorithm 5 `TruncatedProduct`(F,G,d)

Input: $F\in\mathbb{F}[x]^{n\times m}$, $G\in\mathbb{F}[x]^{m\times m}$, $d=(d_1,\ldots,d_m)\in\mathbb{N}^m$

Output: the column-truncated product $(FG)\operatorname{rem} x^d\in\mathbb{F}[x]^{n\times m}$

1: $\gamma\leftarrow$ sum of the degrees of the nonzero columns of F
2: $D\leftarrow\max(|d|,\gamma)$; $\delta\leftarrow\lceil D/m\rceil$; $\ell\leftarrow\lceil\log_2(m)\rceil$
3: write $F=F^{(0)}+\sum_{1\le k<\ell}F^{(k)}x^{2^k\delta}$ with each $F^{(k)}$ in $\mathbb{F}[x]^{n\times m}$, $\deg(F^{(k)})<2^k\delta$ for $1\le k<\ell$, and $\deg(F^{(0)})<2\delta$
4: $R\in\mathbb{F}[x]^{n\times m}\leftarrow(F^{(0)}G)\operatorname{rem} x^d$ ▷ *stores the result*
5: **for** $k\leftarrow 1,\ldots,\ell-1$ **do**
6: $\quad I\leftarrow\{i\in\{1,\ldots,m\}\mid\operatorname{cdeg}(F_{*,i})\ge 2^k\delta\}$
7: $\quad J\leftarrow\{j\in\{1,\ldots,m\}\mid d_j\ge 2^k\delta\}$
8: $\quad e\leftarrow(d_j-2^k\delta)_{j\in J}$
9: $\quad R_{*,J}\leftarrow R_{*,J}+\left((F^{(k)}_{*,I}G_{I,J})\operatorname{rem} x^e\right)x^{2^k\delta}$
10: **end for**
11: **return** R

At Line 9, the multiplication by a power of x is free. The sum consists in adding two matrices with n rows and with column degrees strictly less than $(d_j)_{j\in J_k}$ entry-wise. This costs $O(n|d|)$ operations in $\mathbb{F}$ at each iteration, hence $O(\ell n|d|)$ in total. Using the trivial lower bound on $\mathsf{MM}(\cdot,\cdot)$ shows that this is within the claimed overall cost bound:

$$\begin{aligned}\ell n|d|\le\sum_{k=0}^{\ell-1}nD\le\sum_{k=0}^{\ell-1}\left\lceil 2^k\frac{n}{m}\right\rceil(2^{-k}m)^2 2^k\left\lceil\frac{D}{m}\right\rceil\\ \in O\left(\sum_{k=0}^{\ell-1}\left\lceil 2^k\frac{n}{m}\right\rceil\mathsf{MM}\left(2^{-k}m,2^k\left\lceil\frac{D}{m}\right\rceil\right)\right).\end{aligned}$$

Finally, for the truncated product, we first multiply $F^{(k)}_{*,I_k}G_{I_k,J_k}$ and then truncate. The left matrix in this product has n rows, $\le 2^{-k}m$ columns, and degree $<2^k\delta$. The right matrix has row and columns dimensions both $\le 2^{-k}m$, and sum of column degrees $\le|d|$. The product can be performed by expanding the columns of G_{I_k,J_k} into $\le 2^{1-k}m$ columns all of degree $\le|d|/(2^{-k}m)\le 2^k\delta$, which leads to the complexity $O(\lceil 2^k\frac{n}{m}\rceil\,\mathsf{MM}(2^{-k}m,2^k\delta))$. Summing the latter bound for $k\in\{1,\ldots,\ell-1\}$, and adding the term $k=0$ for Line 4, yields the claimed cost bound.

The final simplified complexity bound follows from the assumptions mentioned in Section 6.1: $\mathsf{MM}(\mu,\delta)$ is in $O(\mu^\omega\,\mathsf{M}(\delta))$, $\mathsf{M}(\cdot)$ is superlinear, and $\omega>2$ implies that the sum $\sum_{0\le k<\ell}2^{k(2-\omega)}$ is bounded by a constant. ☐

REFERENCES

[1] J. Alman, R. Duan, V. V. Williams, Y. Xu, Z. Xu, and R. Zhou. 2024. *More Asymmetry Yields Faster Matrix Multiplication.* arXiv:2404.16349. https://arxiv.org/abs/2404.16349

[2] B. Beckermann, G. Labahn, and G. Villard. 2006. Normal forms for general polynomial matrices. *J. Symbolic Comput.* 41, 6 (2006), 708–737. https://doi.org/10.1016/j.jsc.2006.02.001

[3] R. Duan, H. Wu, and R. Zhou. 2023. Faster Matrix Multiplication via Asymmetric Hashing. In *Proceedings 64th IEEE Symposium on Foundations of Computer Science (FOCS).* 2129–2138. https://doi.org/10.1109/FOCS57990.2023.00130

[4] F. R. Gantmacher. 1960. *The Theory of Matrices. Volume one.* Chelsea Publishing Company. https://bookstore.ams.org/chelgantset

[5] J. von zur Gathen and J. Gerhard. 1999. *Modern computer algebra.* Third edition 2013. Cambridge University Press. https://doi.org/10.1017/CBO9781139856065

[6] M. Giesbrecht. 1995. Nearly Optimal Algorithms for Canonical Matrix Forms. *SIAM J. Comput.* 24, 5 (1995), 948–969. https://doi.org/10.1137/S0097539793252687

[7] S. Gupta, S. Sarkar, A. Storjohann, and J. Valeriote. 2012. Triangular x-basis decompositions and derandomization of linear algebra algorithms over $K[x]$. *J. Symbolic Comput.* 47, 4 (2012), 422–453. https://doi.org/10.1016/j.jsc.2011.09.006

[8] N. Jacobson. 2009. *Basic Algebra I.* Dover Publications Inc. https://store.doverpublications.com/0486471896.html Second Edition W.H. Freeman 1985.

[9] C.-P. Jeannerod, V. Neiger, É. Schost, and G. Villard. 2017. Computing minimal interpolation bases. *J. Symbolic Comput.* 83 (2017), 272–314. https://doi.org/10.1016/j.jsc.2016.11.015

[10] T. Kailath. 1980. *Linear Systems.* Prentice-Hall.

[11] R.E. Kalman. 1963. Mathematical Description of Linear Dynamical Systems. *Journal of the Society for Industrial and Applied Mathematics, Series A: Control* 1, 2 (1963), 152–192. https://doi.org/10.1137/0301010

[12] W. Keller-Gehrig. 1985. Fast algorithms for the characteristic polynomial. *Theoretical computer science* 36 (1985), 309–317. https://doi.org/10.1016/0304-3975(85)90049-0

[13] G. Labahn, V. Neiger, and W. Zhou. 2017. Fast, deterministic computation of the Hermite normal form and determinant of a polynomial matrix. *J. Complexity* 42 (2017), 44–71. https://doi.org/10.1016/j.jco.2017.03.003

[14] S. Lang. 2002. *Algebra (revised third edition).* Springer-Verlag New-York Inc. https://doi.org/10.1007/978-1-4613-0041-0

[15] R. T. Moenck and J. H. Carter. 1979. Approximate algorithms to derive exact solutions to systems of linear equations. In *Proceedings International Symposium on Symbolic and Algebraic Manipulation (Eurosam) (LNCS 72).* 65–73. https://doi.org/10.1007/3-540-09519-5_60

[16] V. Neiger and C. Pernet. 2021. Deterministic computation of the characteristic polynomial in the time of matrix multiplication. *J. Complexity* 67 (2021), 101572. https://doi.org/10.1016/j.jco.2021.101572

[17] V. Neiger and T. X. Vu. 2017. Computing canonical bases of modules of univariate relations. In *Proceedings ISSAC 2017* (Kaiserslautern, Germany). ACM, 357–364. https://doi.org/10.1145/3087604.3087656

[18] C. Pernet and A. Storjohann. 2007. Faster Algorithms for the Characteristic Polynomial. In *Proceedings ISSAC 2007.* ACM, 307–314. https://doi.org/10.1145/1277548.1277590

[19] C. Pernet and A. Storjohann. 2007. *Frobenius form in expected matrix multiplication time over sufficiently large fields.* Technical Report. https://cs.uwaterloo.ca/~astorjoh/cpoly.pdf

[20] A. Storjohann. 2000. *Algorithms for Matrix Canonical Forms.* Ph. D. Dissertation. Institut für Wissenschaftliches Rechnen, ETH-Zentrum, Zurich, Switzerland. https://www.research-collection.ethz.ch/bitstream/handle/20.500.11850/145127/eth-24018-02.pdf

[21] A. Storjohann. 2001. Deterministic computation of the Frobenius form. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science (FOCS).* IEEE, 368–377. https://doi.org/10.1109/SFCS.2001.959911

[22] A. Storjohann. 2003. High-order lifting and integrality certification. *J. Symbolic Comput.* 36, 3-4 (2003), 613–648. https://doi.org/10.1016/S0747-7171(03)00097-X

[23] G. Villard. 1997. Fast Parallel Algorithms for Matrix Reduction to Normal Forms. *Appl. Algebra Eng. Commun. Comput.* 8, 6 (1997), 511–537. https://doi.org/10.1007/s002000050089

[24] V. V. Williams, Y. Xu, Z. Xu, and R. Zhou. 2024. New Bounds for Matrix Multiplication: from Alpha to Omega. In *Proceedings ACM-SIAM Symposium on Discrete Algorithms (SODA).* 3792–3835. https://doi.org/10.1137/1.9781611977912.134

[25] W. Zhou and G. Labahn. 2013. Computing Column Bases of Polynomial Matrices. In *Proceedings ISSAC 2013.* ACM, 379–386. https://doi.org/10.1145/2465506.2465947

[26] W. Zhou, G. Labahn, and A. Storjohann. 2012. Computing Minimal Nullspace Bases. In *Proceedings ISSAC 2012* (Grenoble, France). ACM, 366–373. https://doi.org/10.1145/2442829.2442881

[27] W. Zhou, G. Labahn, and A. Storjohann. 2015. A deterministic algorithm for inverting a polynomial matrix. *J. Complexity* 31, 2 (2015), 162–173. https://doi.org/10.1016/j.jco.2014.09.004

A DECOMPOSITION OF THE SPACE $\mathbb{F}^n$

The following is part of the basic material when studying the behavior of a linear operator A and the decomposition of $\mathbb{F}^n$ into cyclic subspaces [4, Chap. VII], which corresponds to the diagonal matrix form of Smith and the block diagonal form of Frobenius. The same concepts can also be used for decompositions associated with the triangular form of Hermite [23;20, Chap. 9], or more general forms such as column reduced ones [10, Sec. 6.4.6, p. 424].

Existence of a Krylov basis. Let $A \in \mathbb{F}^{n\times n}$ and $U \in \mathbb{F}^{n\times m}$, and for $1 \le j \le m$, let d_j be the first integer such that

$$A^{d_j} u_j \in \mathrm{Span}(u_j, Au_j, \dots, A^{d_j-1}u_j) + \mathrm{Orb}(A, U_{*,1..j-1}). \qquad (10)$$

We prove that the columns of $\mathrm{K}_d(A,U)$ form a basis of $\mathrm{Orb}(A,U)$.

Given a subspace $\mathcal{E} \subseteq \mathbb{F}^n$ invariant with respect to A, we say that two vectors $u, v \in \mathbb{F}^n$ are congruent modulo $\mathcal{E}$ if and only if $u - v \in \mathcal{E}$, and we write $u \equiv v \bmod \mathcal{E}$. For a fixed u, the set of polynomials $p \in \mathbb{F}[x]$ such that $p(A)u \equiv 0 \bmod \mathcal{E}$ is an ideal of $\mathbb{F}[x]$, generated by a monic polynomial which is the minimal polynomial of u modulo $\mathcal{E}$. In particular, if $p(A)u \equiv 0 \bmod \mathcal{E}$, then $q(A)u \equiv 0 \bmod \mathcal{E}$ for all multiples $q \in \mathbb{F}[x]$ of p.

The proof of the characterization in Eq. (10) is by induction on m. For one vector, d_1 is the first integer such that

$$(x^{d_1})(A) = A^{d_1}u_1 \equiv 0 \bmod \mathrm{Span}(u_1, Au_1, \dots, A^{d_1-1}u_1).$$

Therefore all the subsequent vectors $(x^{d_1}x^k)(A) = A^{d_1+k}u_1$ with $k \ge 0$ are zero modulo $\mathrm{K}_{d_1}(A,u_1)$ which thus forms a basis of $\mathrm{Orb}(A,u_1)$. Then assume that the property holds for all U of column dimension $m \ge 1$: $\mathrm{K}_d(A,U)$ is a basis of $\mathrm{Orb}(A,U)$. For $v \in \mathbb{F}^m$, let d_{m+1} be the smallest integer so that $A^{d_{m+1}}v$ is a combination of the previous iterates of v and the vectors in $\mathrm{Orb}(A,U)$. By analogy to the case $m = 1$, all subsequent vectors $A^{d_{m+1}+k}v$ are also linear combinations of the vectors in $\mathrm{Span}(v, Av, \dots, A^{d_{m+1}-1}v) + \mathrm{Orb}(A,U)$. So the latter subspace is $\mathrm{Orb}(A, [U, v])$, and by induction hypothesis the columns of $\mathrm{K}_{d_{m+1}}(A,v)$ and $\mathrm{K}_d(A,U)$ form one of its bases.

Maximal Krylov indices. The tuple d constructed this way is lexicographically maximal so that $\mathrm{K}_d(A,U)$ is a basis of $\mathrm{Orb}(A,U)$. The existence of an ℓ such that $(d'_1, \dots, d'_\ell, \dots d'_m)$ is another suitable tuple with $d'_\ell < d_\ell$ would indeed contradict the fact that d_ℓ corresponds to the smallest linear dependence.

B A SLIGHTLY DIFFERENT ALGORITHM

The matrices $xI - A$ and $I - xA$ considered in Algorithm `MaxIndices` and Algorithm `KrylovMatrix` mirror each other. As we explain below, $xI - A$ could also be used to compute a Krylov matrix. (Algorithm `KrylovMatrix` instead considers $I - xA$ simply for a slightly more convenient presentation.)

Consider a minimal kernel basis $[S^\top \ \ T^\top]^\top$ of $[xI - A \ \ -U]$ as in Algorithm `MaxIndices`. Since $(xI - A)^{-1}U$ is strictly proper and $S = (xI - A)^{-1}UT$, the column degrees in T are greater than those in S, and T is column reduced since the kernel basis is (Lemma 2.1). Let d_j be the degree of the jth column of T, and let $d = (d_1, \dots, d_m)$. We denote by x^d the $m \times m$ diagonal polynomial matrix with diagonal entries $(x^{d_1}, \dots, x^{d_m})$. Substituting $1/x$ for x in $(xI - A)S - UT = 0$, and multiplying on the right by x^d we get

$$(I - xA)S(1/x)x^{d-1} - UT(1/x)x^d = 0.$$

By definition of the d_j's, the right-hand term is a polynomial matrix, say $U\hat{T}$. Since the jth column of S has degree at most $d_j - 1$ (and is zero if $d_j = 0$), the left-hand term is also a polynomial matrix, say $(I - xA)\hat{S}$. It follows that the columns of $[\hat{S}^\mathsf{T} \quad \hat{T}^\mathsf{T}]^\mathsf{T}$ are in the kernel of $[I - xA \quad -U]$, and we can now also verify that they form a basis. Since $[S^\mathsf{T} \quad T^\mathsf{T}]^\mathsf{T}$ is itself a minimal basis, it is irreducible, i.e. has full rank for all finite values of x [10, Thm 6.5-10, p. 458]. Equivalently, S and T are coprime [10, Lem. 6.3-6, p. 379], so there exists V unimodular such that $V[S^\mathsf{T} \quad T^\mathsf{T}]^\mathsf{T} = [I \quad 0]^\mathsf{T}$. It follows that

$$V(1/x) \begin{bmatrix} x\hat{S} \\ \hat{T} \end{bmatrix} = \begin{bmatrix} I \\ 0 \end{bmatrix} x^d.$$

and since V is unimodular, the rank of $[\hat{S}(x_0)^\mathsf{T} \quad \hat{T}(x_0)^\mathsf{T}]^\mathsf{T}$ is full for all values $x_0 \neq 0$ of x. The fact that T is column reduced adds that $\hat{T}(0)$ is invertible, which means that $[\hat{S}^\mathsf{T} \quad \hat{T}^\mathsf{T}]^\mathsf{T}$ is irreducible. By irreducibility ([10, Thm 6.5-10, p. 458] again, here after column reduction), we get as announced that $[\hat{S}^\mathsf{T} \quad \hat{T}^\mathsf{T}]^\mathsf{T}$ is a kernel basis of $[I - xA \quad -U]$.

If A is nonsingular, then, in a similar way to what we saw with Lemma 3.3, we can prove that $T(0)$ is invertible. In this case it follows that $\hat{T}$ is column reduced, and that $[\hat{S}^\mathsf{T} \quad \hat{T}^\mathsf{T}]^\mathsf{T}$ is a minimal kernel basis of $[I - xA \quad -U]$. Minimality is not guaranteed if A is singular.

We see that Algorithm `KrylovMatrix` could be modified using the same kernel basis as in Algorithm `MaxIndices`, and by inserting an instruction to work with $\hat{T}$ afterwards. Since $\hat{T}(0)$ si invertible (T is column reduced), the fact that this basis may not be minimal is not a problem for the power series expansion and the subsequent steps in Algorithm `KrylovMatrix`. Also, since $|\mathrm{cdeg}(\hat{T})| \leq |\mathrm{cdeg}(T)| \leq n$, the cost bound in Lemma 3.2 is not affected. In particular, when using Algorithm `MaxIndices` and Algorithm `KrylovMatrix` in succession as in Algorithm `MaxKrylovBasis`, this would lead to computing only one kernel basis instead of two.