Department of Computer Science, KU Leuven, B-3001 Heverlee, Belgium

A randomized algorithm for testing nonsingularity of structured matrices with an application to asserting nondefectivity of Segre varieties

NICK VANNIEUWENHOVEN†, *Raf Vandebril*‡ *and Karl Meerbergen*§

[Received on 23 May 2016]

Tensors admitting an expression as the sum of at most s rank one tensors can be considered points of the sth order secant variety of a Segre variety, yet the most basic invariant of this variety—its dimension—is not yet fully understood. A conjecture was nevertheless proposed by Abo *et al.* (2009, *Trans. Amer. Math. Soc.*, **361**, 767–792), proved to be correct for $s \leqslant 6$. We propose a numerical randomized algorithm for testing whether a mathematically exact and structured matrix is singular requiring only the availability of an approximate matrix-vector product. Using this method we test whether the aforementioned conjecture is true. The proposed method requires several orders of magnitude less memory than approaches based on symbolic arithmetic, thus greatly increasing the number of varieties that can be handled. Our experiments establish that the Segre varieties $\mathbb{PC}^{n_1} \times \mathbb{PC}^{n_2} \times \cdots \times \mathbb{PC}^{n_d}$ embedded in $\mathbb{PC}^{n_1 \times \cdots \times n_d}$ with expected generic rank $s \leqslant 55$ obey the conjecture; the probability that a defective variety is incorrectly classified as nondefective is less than 10^{-55}.

Keywords: Algebraic geometry; dimensions of secant varieties of Segre varieties; randomized algorithm; numerical analysis.

1. Introduction

(32) introduced an expression of a multidimensional array $\mathscr{A} \in \mathbb{C}^{n_1 \times n_2 \times \cdots \times n_d}$, assuming without loss of generality that $n_1 \geqslant n_2 \geqslant \cdots \geqslant n_d \geqslant 2$, as a sum of rank one tensors:

$$\mathscr{A} = \sum_{i=1}^{s} \lambda_i \mathbf{a}_i^1 \otimes \mathbf{a}_i^2 \otimes \cdots \otimes \mathbf{a}_i^d, \tag{1.1}$$

with $\lambda_i \in \mathbb{C}$, $\mathbf{a}_i^k \in \mathbb{C}^{n_k}$ of unit Euclidean norm, $\otimes$ the tensor product, and where it is assumed s is minimal. An array admitting the rank s decomposition in (1.1) may be parameterized by the λ_i and

$$A^{(k)} := \begin{bmatrix} \mathbf{a}_1^k & \mathbf{a}_2^k & \dots & \mathbf{a}_s^k \end{bmatrix}, \quad \text{for } 1 \leqslant k \leqslant d.$$

In the context of algebraic geometry, (1.1) may be considered a parameterization of a point on the sth order *secant variety* $\sigma_s(\mathscr{S})$ of the Segre variety $\mathscr{S} = \mathbb{PC}^{n_1} \times \mathbb{PC}^{n_2} \times \cdots \times \mathbb{PC}^{n_d} \subseteq \mathbb{PC}^{n_1 n_2 \cdots n_d}$, after choosing bases for the $(n_k - 1)$-dimensional *projective spaces* $\mathbb{PC}^{n_k}$. It is common to work over projective spaces, hereby considering $\mathscr{A} = \lambda \mathscr{A}$ if $\lambda \neq 0$, because the quantities of interest—e.g., degree, dimension, and rank—are invariant to scaling. Quotienting out this scalar factor will sometimes be referred to as "projectivization". The secant variety $\sigma_s(\mathscr{S})$ is formally defined as the Zariski closure of

†Corresponding author. Email: nick.vannieuwenhoven@cs.kuleuven.be
‡Email: raf.vandebril@cs.kuleuven.be
§Email: karl.meerbergen@cs.kuleuven.be

the union of the linear span of s points on $\mathscr{S}$ (36):

$$\sigma_s(\mathscr{S}) = \overline{\bigcup_{p_1,p_2,\ldots,p_s\in\mathscr{S}} \operatorname{span}(p_1,p_2,\ldots,p_s)},$$

i.e., the Zariski closure of all tensors satisfying (1.1). Letting

$$\Pi := \prod_{k=1}^{d} n_i \quad \text{and} \quad \Sigma := \sum_{k=1}^{d} n_i,$$

the *expected dimension* of $\sigma_s(\mathscr{S})$ is $\min\{s(\Sigma-d+1)-1,\Pi-1\}$; i.e., the minimum of s times the dimension of the Segre variety $\mathscr{S}$, and the dimension of the ambient space. Unfortunately, the dimension of $\sigma_s(\mathscr{S})$ can be strictly less than this expected dimension, in which case we call $\mathscr{S}$ *s-defective*. A Segre variety is called *defective* if it is s-defective for some s. The defective secant varieties are presently not known in all cases, notwithstanding their intrinsic importance; for instance, a defective sth order secant variety implies that a generic rank s tensor does not admit a unique expression of type (1.1).

Defects of secant varieties of algebraic varieties have been studied classically, and recently much progress has been made for Segre (4; 10; 12; 15; 16; 37; 48), Segre–Veronese (3; 2; 11; 14), Grassmann (5; 13; 17; 39), and subspace (54) varieties. The defective secant varieties of the Veronese variety were conjectured at least 150 years ago (9), and it was proven to be correct by **(author?)** (6). For the Segre variety, only a few defective cases are known, which led to

Conjecture 1.1 ((4)) The Segre variety $\mathscr{S} = \mathbb{PC}^{n_1}\times\mathbb{PC}^{n_2}\times\cdots\times\mathbb{PC}^{n_d}$ is nondefective, unless one of the following holds:

1. $n_1 > 1+\prod_{k=2}^{d} n_k - \sum_{k=2}^{d}(n_k-1)$, i.e., $\mathscr{S}$ is unbalanced,
2. $\mathscr{S} = \mathbb{PC}^4\times\mathbb{PC}^4\times\mathbb{PC}^3$,
3. $\mathscr{S} = \mathbb{PC}^{2n+1}\times\mathbb{PC}^{2n+1}\times\mathbb{PC}^3$, $n\in\mathbb{N}$,
4. $\mathscr{S} = \mathbb{PC}^{n}\times\mathbb{PC}^{n}\times\mathbb{PC}^2\times\mathbb{PC}^2$, $n\in\mathbb{N}$.

(author?) (4) verified that the sth order secant varieties of the Segre varieties are not defective when $s\leqslant 6$, except for the known defective cases in the above list. They proposed an inductive proof strategy suitable for automated computer processing, manually proving a number of base cases from which the induction can start. In theory, this approach can prove the nondefectivity of any Segre variety; however, the core reduction process employed is combinatorial in nature, thus limiting the practical applicability of this method for large ranks and large dimensions.

An approach complementary to the inductive procedure of **(author?)** (4) is also founded on Terracini's lemma (50; 55; 36); this classic lemma provides a local characterization of the dimensions of secant varieties:

THEOREM 1.2 (Terracini's Lemma) Let $\mathscr{S}\subseteq\mathbb{PC}^N$ be a Segre variety, and let $p_1,p_2,\ldots,p_s\in\mathscr{S}$ be generic points. Let $\mathbb{T}_p\mathscr{A}$ denote the projectivized tangent space of an algebraic variety $\mathscr{A}\subseteq\mathbb{PC}^N$ at $p\in\mathscr{A}$. If p is a generic point in the linear span $\langle p_1,p_2,\ldots,p_s\rangle$, then

$$\dim\sigma_s(\mathscr{S}) = \dim\mathbb{T}_p\sigma_s(\mathscr{S}) = \dim\langle\mathbb{T}_{p_1}\mathscr{S},\mathbb{T}_{p_2}\mathscr{S},\ldots,\mathbb{T}_{p_s}\mathscr{S}\rangle.$$

A set of s *witness* points on the Segre variety, and one generic[1] point in their span where the dimension of the tangent space is maximal are sufficient to *prove* that the sth order secant variety has the expected dimension. This leads to a well-known probabilistic algorithm to check whether the sth order secant variety of the Segre variety is nondefective; see, e.g., **(author?)** (1). It randomly selects s points on the Segre variety and verifies whether the rank of the matrix representing the tangent space is the expected one using symbolic arithmetic. This projectivized tangent space to $\mathscr{S} = \mathbb{PC}^{n_1} \times \mathbb{PC}^{n_2} \times \cdots \times \mathbb{PC}^{n_d}$ at $p_s = v_s^1 \otimes v_s^2 \otimes \cdots \otimes v_s^d$ is given by (4; 36):

$$\mathbb{T}_{p_s}\mathscr{S} := \mathbb{PC}^{n_1} \otimes v_s^2 \otimes \ldots \otimes v_s^d + v_s^1 \otimes \mathbb{PC}^{n_2} \otimes v_s^3 \otimes \cdots \otimes v_s^d + \cdots + v_s^1 \otimes \cdots \otimes v_s^{d-1} \otimes \mathbb{PC}^{n_d}. \tag{1.2}$$

One then chooses bases for $\mathbb{PC}^{n_i}$ and parameterizes point p_s by $\mathbf{a}_s^1 \otimes \mathbf{a}_s^2 \otimes \cdots \otimes \mathbf{a}_s^d$, where $\mathbf{a}_s^k \in \mathbb{C}^{n_k}$ is of norm unity and $\otimes$ denotes the Kronecker product. The parameters may be chosen freely without affecting the correctness of the procedure;[2] e.g., they can be drawn independently and identically distributed (i.i.d.) from a real standard normal distribution, *which we will assume throughout the paper*. The span of the kth linear space in (1.2) equals the projectivization of the column span of the $\Pi \times n_k$ matrix

$$T_s^k := \mathbf{a}_s^1 \otimes \cdots \otimes \mathbf{a}_s^{k-1} \otimes I_{n_k} \otimes \mathbf{a}_s^{k+1} \otimes \cdots \otimes \mathbf{a}_s^d, \tag{1.3}$$

having chosen the standard basis for $\mathbb{R}^{n_k}$ for simplicity; I_{n_k} denotes the identity matrix of order n_k. By Terracini's lemma, the span of $\mathbb{T}_p\sigma_s(\mathscr{S})$ is given by the projectivization of the column span of

$$\mathscr{T} := \begin{bmatrix} T^1 & T^2 & \cdots & T^d \end{bmatrix}, \quad \text{where} \quad T^k := \begin{bmatrix} T_1^k & T_2^k & \cdots & T_s^k \end{bmatrix}. \tag{1.4}$$

Thus, the dimension of the tangent space equals the rank of the above matrix minus one. If the latter equals the expected dimension, i.e., if $\operatorname{rank}(\mathscr{T}) = \min\{s(\Sigma - d + 1), \Pi\}$, then we have found a set of witness points. Otherwise, no conclusions can be drawn; the variety may be defective, or the chosen points are not generic. Letting

$$r = \left\lfloor \frac{\Pi}{\Sigma - d + 1} \right\rfloor \quad \text{and} \quad R = \left\lceil \frac{\Pi}{\Sigma - d + 1} \right\rceil, \tag{1.5}$$

it suffices to check that the rth and Rth order secant varieties have the expected dimension to show that the Segre variety is *nondefective* (4). One says that the Segre variety is *subabundant* if the rth order secant variety has the expected dimension, and, similarly, the Segre variety is *superabundant* if the Rth order secant variety fills the ambient space. Note that $\mathscr{T}$ and T^k both depend on the number of points s; however, *in the remainder, these matrices will always be considered for* $s = r$.

The above algorithm, which we call the *classic algorithm*, can be implemented in a computer algebra system using exact arithmetic; see, e.g., (1). The main disadvantage is the cost of symbolic calculation, in terms of memory and execution time, which is not constant but increases with the complexity of the symbolic expression. Floating-point arithmetic, on the other hand, offers a constant memory and execution time cost, and is efficiently implemented on modern computer systems; however, its obvious disadvantage is the loss of exactness. **(author?)** (19) considered the classic algorithm in floating-point arithmetic before, but they do not analyze the effect of the inexact computations.[3] In fact, it is presently

[1] A small remark concerning the use of "genericity" in algebraic geometry is in order: a property is called generic if it holds everywhere except, possibly, on a set of zero measure (in the Zariski topology). This implies that if one samples a point from this set according to an absolutely continuous probability density function, then a generic point is obtained with probability one.

[2] Choosing the subset too small may entail that a generic point of the variety may never be sampled.

[3] We thank an anonymous reviewer for the observation that **(author?)** (18) are probably the first to consider an algorithm of this type—they employed it to assess nondefectivity of the secants of Veronese varieties.

not understood whether the mathematically exact dimension can be recovered from inexact numerical calculations. **(author?)** (8), for instance, state:

> "In a floating-point setting, one may compute the numerical rank of [the numerical approximation of $\mathscr{T}$] by counting the number of singular values larger than some tolerance ε. Thus the numerical rank is a function of ε. This raises the natural question: *How to choose* ε*?* A general precise answer to this question is both unclear and application-dependent."

This paper describes the first reliable implementation of the classic randomized algorithm for testing nondefectivity of Segre varieties in finite precision floating-point arithmetic. Its contribution is threefold: first, the aforementioned difficulties in a straightforward numerical implementation are circumvented by considering a *probabilistic method* to determine the rank of a matrix; second, the probability of failure can be decreased geometrically to zero with *fixed memory demands*; and, third, it is vastly more efficient than symbolic arithmetic while offering a quantifiable degree of certainty.

The core algorithm in this paper is a randomized algorithm well-suited for an implementation using floating-point arithmetic for testing whether a structured matrix is nonsingular. It is assumed that the parameters describing the structured matrix are known exactly, and that an approximate matrix-vector product can be computed efficiently. This is the case when computing the dimensions of certain algebraic varieties; e.g., secant varieties of Segre, Segre-Veronese, Grassmann, and subspace varieties. We shall illustrate the use of the proposed method for testing whether Segre varieties are nondefective. The proposed method might incorrectly conclude that a defective variety is nondefective only with small and quantifiable probability. We stress that such failures do *not* originate from roundoff errors in floating-point arithmetic, but rather from an improbable choice of random starting vectors. This probability of failure can be decreased geometrically by executing the algorithm multiple times.

In the next section, we propose a randomized algorithm using only matrix-vector products for testing nonsingularity of matrices. Section 3 discusses a structure-exploiting and memory efficient algorithm for testing nondefectivity of Segre varieties based on the algorithm of Section 2. Numerical experiments are presented in Section 4. Section 5 features a discussion of concerns related to alternative strategies for verifying nondefectivity. Conclusions are drawn in Section 6.

Concerning notation, the following conventions apply throughout. Matrices are typeset in uppercase (A, B), vectors in a bold face $(\mathbf{a}, \mathbf{b})$. The tensor product is denoted by $\otimes$; this notation is also used for the Kronecker product, which operates on a parameterization. The transpose of a matrix is denoted by $\cdot^T$. The Hadamard, or componentwise, product of matrices is defined as $(A \odot B)_{ij} = a_{ij} b_{ij}$. The Euclidean inner product of $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ is denoted by $\langle \mathbf{a}, \mathbf{b} \rangle$, which immediately defines the corresponding norm $\|\cdot\|$.

2. A randomized matrix-free algorithm for testing nonsingularity of matrices

In this section, we propose a numerical algorithm for probabilistically detecting whether a matrix $A \in \mathbb{R}^{m \times n}$, $m \geqslant n$, is of full rank. The algorithm we propose assumes that the matrix A is not given explicitly but rather accommodates only an efficient matrix-vector product; such is often the case for sparse and structured matrices. We assume that direct methods for computing rank-revealing decompositions, such as a singular value decomposition, are infeasible due to memory considerations.

Here, we consider a strategy for testing whether $A \in \mathbb{R}^{m \times n}$, $m \geqslant n$, is of full column rank, founded on verifying whether $A\mathbf{x} = 0$ admits only the trivial solution. Note that it is theoretically clear that $A^T A$ is of full rank iff A is of full rank, and that $A^T A$ is a symmetric positive semidefinite (spsd) matrix;[4]

[4]Solving the system with $A^T A$ may pose challenges for Krylov subspace methods; this issue is elaborated upon in Section 5.

Algorithm 1: CONJUGATEGRADIENT(A, $\mathbf{b}$, $\mathbf{x}^{(0)}$).

input : Spsd matrix $A \in \mathbb{R}^{n\times n}$, right hand side $\mathbf{b} \in \text{range}\,(A)$, and a starting vector $\mathbf{x}^{(0)}$.
output: The CG solution of $A\mathbf{x} = \mathbf{b}$ with starting vector $\mathbf{x}^{(0)}$.
$\mathbf{r}^{(0)} \leftarrow \mathbf{b} - A\mathbf{x}^{(0)}$
$\mathbf{p}^{(0)} \leftarrow \mathbf{r}^{(0)}$
for $k \leftarrow 1$ *to* n **do**
  $\gamma_{k-1} \leftarrow \|\mathbf{r}^{(k-1)}\|^2 / \langle \mathbf{p}^{(k-1)}, A\mathbf{p}^{(k-1)} \rangle$
  $\mathbf{x}^{(k)} \leftarrow \mathbf{x}^{(k-1)} + \gamma_{k-1}\mathbf{p}^{(k-1)}$
  $\mathbf{r}^{(k)} \leftarrow \mathbf{r}^{(k-1)} - \gamma_{k-1}A\mathbf{p}^{(k-1)}$
  $\beta_k \leftarrow \|\mathbf{r}^{(k)}\|^2 / \|\mathbf{r}^{(k-1)}\|^2$
  $\mathbf{p}^{(k)} \leftarrow \mathbf{r}^{(k)} + \beta_k\mathbf{p}^{(k-1)}$
end

therefore, we may assume without loss of generality that A is spsd. The key idea consists of employing an *iterative* Krylov method for solving a linear system $A\mathbf{x} = \mathbf{b}$, with $\mathbf{b} \in \text{range}\,(A)$; this comprises generating a sequence of nested Krylov spaces

$$\mathscr{K}_k(A, \mathbf{r}) := \text{range}\left(\mathbf{r}, A\mathbf{r}, A^2\mathbf{r}, \ldots, A^{k-1}\mathbf{r}\right) \subseteq \text{range}\,(A)\,,$$

with $\mathbf{r} = \mathbf{b} - A\mathbf{x}^{(0)}$ and $\mathbf{x}^{(0)}$ any *starting vector*, and choosing an approximation $\mathbf{x}^{(k)}$ from the affine space $\mathbf{x}^{(0)} + \mathscr{K}_k(A, \mathbf{r})$; see, e.g., **(author?)** (44). The various Krylov methods differ in the way the approximation $\mathbf{x}^{(k)}$ is chosen; one consists of choosing $\mathbf{b} - A\mathbf{x}^{(k)}$ orthogonal to $\mathscr{K}_k(A, \mathbf{r})$, where orthogonality with respect to the Euclidean inner product is implied. This yields the conjugate gradient (CG) method from (28); it is presented as Algorithm 1. It is well-known that, in exact arithmetic, this algorithm ignores the null space of A, which can be exploited to verify whether $A\mathbf{x} = 0$ admits nontrivial solutions by choosing a good starting vector:[5]

LEMMA 2.1 Let $A \in \mathbb{R}^{n\times n}$ be an spsd matrix and $\mathbf{x}^{(0)} \in \mathbb{R}^n$ a vector whose components were drawn i.i.d. from a standard normal distribution. Let $\mathbf{x}^*$ be the conjugate gradient solution to $A\mathbf{x} = 0$ with starting vector $\mathbf{x}^{(0)}$. If A is nonsingular then $\|\mathbf{x}^*\| = 0$ deterministically; if A is singular then $\|\mathbf{x}^*\| = 0$ with probability zero, and $\|\mathbf{x}^*\| > 0$ with probability one.

Proof. As A is real symmetric, it admits an eigenvalue decomposition $A = VDV^T$, with $V \in \mathbb{R}^{n\times n}$ orthogonal and $D \in \mathbb{R}^{n\times n}$ a diagonal matrix whose diagonal elements are sorted in order of decreasing magnitude. We may then write

$$\mathbf{x}^{(0)} = VV^T\mathbf{x}^{(0)} = \sum_{i=1}^{s} \langle \mathbf{x}^{(0)}, \mathbf{v}_i \rangle \mathbf{v}_i + \sum_{i=s+1}^{n} \langle \mathbf{x}^{(0)}, \mathbf{v}_i \rangle \mathbf{v}_i$$

where s is the rank of A. Because the components of $\mathbf{x}^{(0)}$ were drawn statistically independent from A and because Gaussian distributions are invariant under orthogonal transformations (20, §4.2), it follows

[5] We are convinced that this lemma is known to the experts, but were unable to find a reference.

that $\langle \mathbf{x}^{(0)}, \mathbf{v}_i \rangle \sim N(0,1)$, i.e., they are also standard normally distributed. Consequently, if $s \neq n$, $\mathbf{x}^{(0)}$ has a nonzero coefficient with respect to $\mathbf{v}_i$, $s+1 \leqslant i \leqslant n$, with probability one. In exact arithmetic, Algorithm 1 produces a solution $\mathbf{x}^*$ such that $A\mathbf{x}^* = 0$ when it is applied to the (possibly singular) linear system $A\mathbf{x} = 0$. By induction one can prove that $\mathbf{r}^{(k)} \in \text{range}\,(A)$ and $\mathbf{p}^{(k)} \in \text{range}\,(A)$ for all $0 \leqslant k \leqslant n$. Therefore, $\mathbf{p}^{(k)} := \sum_{i=1}^{s} \phi_i^{(k)} \mathbf{v}_i$, and we find by definition of the CG algorithm that

$$\mathbf{x}^{(k)} = \sum_{i=1}^{s} \Big(\langle \mathbf{x}^{(0)}, \mathbf{v}_i \rangle + \sum_{j=1}^{k} \gamma_{j-1} \phi_i^{(j-1)} \Big) \mathbf{v}_i + \sum_{i=s+1}^{n} \langle \mathbf{x}^{(0)}, \mathbf{v}_i \rangle \mathbf{v}_i,$$

i.e., components of $\mathbf{x}^{(0)}$ in the null space of A are unaffected by the CG algorithm. Then, we have:

$$\mathbf{x}^* = \mathbf{x}^{(n)} = \sum_{i=s+1}^{n} \langle \mathbf{x}^{(0)}, \mathbf{v}_i \rangle \mathbf{v}_i \quad \text{and} \quad \|\mathbf{x}^*\|^2 = \sum_{i=s+1}^{n} \langle \mathbf{x}^{(0)}, \mathbf{v}_i \rangle^2$$

because $A\mathbf{x}^* = 0$. The squared norm of the CG solution $\mathbf{x}^*$ thus obeys the chi-squared distribution with $n-s$ degrees of freedom. The thesis follows. □

In exact arithmetic, the above lemma allows us to distinguish, with probability one, between singular and nonsingular matrices: we simply check the norm of the solution. The situation is more involved in finite precision floating-point arithmetic as $\mathbf{x}^*$ will never be exactly zero due to roundoff errors. However, if both the norm of the inexact approximation and the roundoff errors are small, the true solution $\mathbf{x}^*$ must also have a small norm:[6]

LEMMA 2.2 Let A, $\mathbf{x}^{(0)}$, and $\mathbf{x}^*$ be as in Lemma 2.1. Let $\underline{\mathbf{x}}^{(m)}$ be the approximate solution to $A\mathbf{x} = 0$ obtained by applying the CG method in inexact arithmetic with starting vector $\mathbf{x}^{(0)}$ for m steps. Let $W \in \mathbb{R}^{n \times t}$ be a matrix whose orthonormal columns span the t-dimensional null space of A, then we have

$$\|\mathbf{x}^*\| = \|W^T \mathbf{x}^{(0)}\| \leqslant \|\underline{\mathbf{x}}^{(m)}\| + \|W^T (\mathbf{x}^{(0)} - \underline{\mathbf{x}}^{(m)})\| =: \eta^{(m)} + \chi^{(m)}.$$

Proof. The equality on the left-hand side of the inequality follows from the observation regarding $\mathbf{x}^* = \mathbf{x}^{(n)}$ in the proof of Lemma 2.1. The inequality is due to $\|W^T \mathbf{x}^{(0)}\| \leqslant \|W^T \underline{\mathbf{x}}^{(m)}\| + \|W^T (\mathbf{x}^{(0)} - \underline{\mathbf{x}}^{(m)})\|$. □

This lemma is useful because we will show that $\chi^{(m)}$ can be computed with a three-term recurrence while performing the CG method; hence, the lemma entails a computable bound on $\|\mathbf{x}^*\|$ that we will combine with a variant of Lemma 2.1 to obtain an *a posteriori* likelihood that A is singular.

A three-term recurrence computing $\chi^{(m)}$ is obtained from a forward error analysis (53) of Algorithm 1. While many works have analyzed the behavior of the CG method in a numerical setting (see **(author?)** (40) for a good overview) from the viewpoint of convergence (23; 24; 41; 46), error estimation and reliable stopping criteria (22; 47), and loss of orthogonality (42), none appear to be immediately applicable to the present problem as these analyses ignore second-order error terms. Therefore, we include a brief analysis, similar to (47, Section 7), but without ignoring the second-order terms. To proceed, we will henceforth assume the standard model of floating-point arithmetic (31; 53):

$$\text{fl}\,(x \,\text{op}\, y) = (1+\delta)(x \,\text{op}\, y), \quad \text{with } |\delta| \leqslant \varepsilon, \quad \text{and where op} = +, -, \cdot, /; \tag{2.1}$$

herein, ε is the unit roundoff. The IEEE floating-point arithmetic standard satisfies this model and is implemented by most modern desktop computer systems, including the test setup employed in the numerical experiments in Section 4. The presented analysis will provide only a rough upper bound on

[6] We thank an anonymous referee for the presented proof of this lemma, which is much clearer than our original one.

the actual computation errors. We abide by the convention that $\underline{a} := \mathrm{fl}(a)$ denotes the actual number stored by the computer system, and that $|\mathbf{a}|$ denotes the vector of absolute values of $\mathbf{a}$. Whenever a matrix or vector appears in an inequality with a scalar, the inequality should be interpreted componentwise.

We may now state the three-term recurrence for computing $\chi^{(m)}$:

LEMMA 2.3 Let A, W, $\mathbf{x}^{(0)}$, $\underline{\mathbf{x}}^{(m)}$, and $\mathbf{x}^*$ be as in Lemma 2.2. Let $\underline{\gamma}_k$, $\underline{\beta}_k$, $\underline{\mathbf{x}}^{(k)}$, $\underline{\mathbf{p}}^{(k)}$, $\underline{\mathbf{r}}^{(k)}$ be the floating-point approximations of the matching variables in Algorithm 1 in step k, and $\underline{\mathbf{q}}^{(k-1)} = \mathrm{fl}(A\underline{\mathbf{p}}^{(k-1)})$. Let

$$\chi^{(k)} := \|W^T(\underline{\mathbf{x}}^{(k)} - \mathbf{x}^{(0)})\|, \quad \rho^{(k)} := \|W^T\underline{\mathbf{r}}^{(k)}\|, \quad \text{and } \phi^{(k)} := \|W^T\underline{\mathbf{p}}^{(k)}\|,$$

then these norms satisfy the three-term recurrence

$$\begin{aligned}
\chi^{(k)} &\leqslant \chi^{(k-1)} + |\underline{\gamma}_{k-1}|\phi^{(k-1)} + \varepsilon\left\|(2+\varepsilon)\,|\underline{\gamma}_{k-1}|\,|\underline{\mathbf{p}}^{(k-1)}| + |\underline{\mathbf{x}}^{(k-1)}|\right\|;\\
\rho^{(k)} &\leqslant \rho^{(k-1)} + \varepsilon\left\|(2+\varepsilon)\,|\underline{\gamma}_{k-1}|\,|\underline{\mathbf{q}}^{(k-1)}| + |\underline{\mathbf{r}}^{(k-1)}| + |\underline{\gamma}_{k-1}|\,\varepsilon^{-1}\delta(A,\underline{\mathbf{p}}^{(k-1)})\mathbf{1}\right\|;\\
\phi^{(k)} &\leqslant \rho^{(k)} + |\underline{\beta}_k|\,|\phi^{(k-1)}| + \varepsilon\left\|(2+\varepsilon)\,|\underline{\beta}_k|\,|\underline{\mathbf{p}}^{(k-1)}| + |\underline{\mathbf{r}}^{(k)}|\right\|;
\end{aligned}$$

where $\mathbf{1}$ is a vector of all ones, and $\delta(\cdot,\cdot)$ is a scalar upper bound on the forward error of the matrix-vector product $A\underline{\mathbf{p}}^{(k-1)}$:

$$\left|A\underline{\mathbf{p}}^{(k-1)} - \mathrm{fl}\left(A\underline{\mathbf{p}}^{(k-1)}\right)\right| \leqslant \delta(A,\underline{\mathbf{p}}^{(k-1)}).$$

The initial values of the recurrence are:

$$\chi^{(0)} = 0 \quad \text{and} \quad \phi^{(0)} = \rho^{(0)} = \sqrt{n}\cdot\delta(A,\mathbf{x}^{(0)}).$$

Proof. Remark that $\chi^{(k)}$, $\rho^{(k)}$ and $\phi^{(k)}$ can be interpreted as the norm of the components *generated in the null space due to the numerical execution of the CG method*. In exact arithmetic, $\chi^{(k)} = \rho^{(k)} = \phi^{(k)} = 0$; however, in floating-point arithmetic they can become nonzero. This theorem bounds their magnitude.

From a standard forward analysis we find:

$$\underline{\mathbf{x}}^{(k)} = \underline{\mathbf{x}}^{(k-1)} + \underline{\gamma}_{k-1}\underline{\mathbf{p}}^{(k-1)} + \mathbf{e}^x; \tag{2.2}$$

$$\underline{\mathbf{r}}^{(k)} = \underline{\mathbf{r}}^{(k-1)} - \underline{\gamma}_{k-1}A\underline{\mathbf{p}}^{(k-1)} + \mathbf{e}^r; \tag{2.3}$$

$$\underline{\mathbf{p}}^{(k)} = \underline{\mathbf{r}}^{(k)} + \underline{\beta}_k\underline{\mathbf{p}}^{(k-1)} + \mathbf{e}^p; \tag{2.4}$$

where the forward errors, without ignoring the higher-order terms in ε, can be bounded by:

$$\begin{aligned}
\|\mathbf{e}^x\| &\leqslant \varepsilon\||\underline{\mathbf{x}}^{(k-1)}| + (2+\varepsilon)|\underline{\gamma}_{k-1}||\underline{\mathbf{p}}^{(k-1)}|\|;\\
\|\mathbf{e}^r\| &\leqslant \varepsilon\|(2+\varepsilon)|\underline{\gamma}_{k-1}||\underline{\mathbf{q}}^{(k-1)}| + |\underline{\mathbf{r}}^{(k-1)}| + |\underline{\gamma}_{k-1}|\varepsilon^{-1}\delta(A,\underline{\mathbf{p}}^{(k-1)})\mathbf{1}\|;\\
\|\mathbf{e}^p\| &\leqslant \varepsilon\||\underline{\mathbf{r}}^{(k)}| + (2+\varepsilon)|\underline{\beta}_k||\underline{\mathbf{p}}^{(k-1)}|\|.
\end{aligned}$$

Note that the first-order terms in equations (7.9)–(7.11) in **(author?)** (47) can be obtained from the above bounds; the bounds presented here are a bit tighter. A detailed derivation of (2.2)–(2.4) was presented in the proof of Theorem 3 in (52).

Consider now (2.2), then

$$W^T(\underline{\mathbf{x}}^{(k)} - \mathbf{x}^{(0)}) = W^T(\underline{\mathbf{x}}^{(k-1)} - \mathbf{x}^{(0)}) + \underline{\gamma}_{k-1}W^T\underline{\mathbf{p}}^{(k-1)} + W^T\mathbf{e}^x, \tag{2.5}$$

so that taking norms, applying the triangle inequality to (2.5), and bounding $\|W^T\mathbf{e}^x\| \leqslant \|\mathbf{e}^x\|$ yields the recursion for $\chi^{(m)}$. The two other recurrences are likewise obtained by multiplying (2.4), respectively (2.3), with W^T, taking norms and applying the triangle inequality. Note that the second term on the right-hand side of (2.3) becomes zero when multiplied with W^T, because the term is in the range of A.

The initial values of the three-term recurrence are similarly found, concluding the proof. □

Assuming the matrix is singular with a t-dimensional null space, we may compute an upper bound on the *a posteriori* likelihood that $\|\mathbf{x}^*\|$ was obtained, given the $\eta^{(m)}$ and $\chi^{(m)}$ obtained by running the CG algorithm for m steps. By definition and Lemma 2.1, this likelihood is defined as $P(z \leqslant \|\mathbf{x}^*\|^2)$ where z is chi-squared distributed with t degrees of freedom. Unfortunately, we do not know the value of t, only that $t \geqslant 1$. Consider hence the following: if W is an orthogonal basis for the null space of A,

$$|\langle \mathbf{x}^{(0)}, \mathbf{w}\rangle| \leqslant \|W^T\mathbf{x}^{(0)}\| = \|\mathbf{x}^*\| \quad \text{for every } \mathbf{w} \in \text{range}(W) \text{ with } \|\mathbf{w}\| = 1.$$

However, as was already mentioned in the proof of Lemma 2.1, $z := \langle \mathbf{x}^{(0)}, \mathbf{w}\rangle$ is standard normally distributed, while the above combined with Lemma 2.2 and 2.3 concurrently asserts that its magnitude must be bounded by $\upsilon^{(m)} := \eta^{(m)} + \chi^{(m)}$. The *a posteriori* likelihood that $\|\mathbf{x}^*\| \leqslant \upsilon^{(m)}$ given that A is singular is then bounded from above by $P(|z| \leqslant \upsilon^{(m)})$ where $z \sim N(0,1)$. For the sake of simplicity, we will bound this probability from above by $\min\{1, \upsilon^{(m)}\}$. It is important to realize that this *a posteriori* likelihood yields information about the exact starting vector $\mathbf{x}^{(0)}$.

ALGORITHM S. The matrix-free randomized algorithm we propose for testing whether a matrix is nonsingular can be summarized as follows. Bearing the above discussion in mind, compute ℓ approximate CG solutions $\mathbf{x}_i^{(m_i)}$ with the respective starting vectors $\mathbf{x}_i^{(0)}$ (whose components are standard normal i.i.d.). Let $\omega < 1$ be a user-chosen threshold. If A is assumed to be singular and *each* of the obtained upper bounds $\upsilon_i^{(m_i)} \leqslant \omega$, the probability of having sampled these starting vectors is at most $\omega^\ell \ll 1$ because of the statistical independence of these vectors. By suitably choosing ω and ℓ, the user can shrink this probability such that one will have to consider it an overwhelming amount of counter-evidence to the hypothesis that A is singular. If, in this case, one concludes that A is nonsingular, the probability of incorrectly asserting that a singular matrix is nonsingular is bounded by ω^ℓ, which the user can make as small as desired. Summarizing, the proposed algorithm will reject the hypothesis that A is singular whenever $\upsilon_i^{(m_i)} \leqslant \omega$ for *all* $i = 1, \ldots, \ell$. If, on the other hand, $\upsilon_i^{(m_i)} > \omega$ for *any* i, the algorithm conservatively claims that it is *undecided*: there is insufficient evidence to reject the initial hypothesis.

It should be stressed that *the failure probability ω^ℓ cannot be interpreted as the probability that A is singular*. Such an interpretation is nonsensical because A is either deterministically singular or nonsingular. The correct interpretation is that when one applies the proposed algorithm $\omega^{-\ell}$ times to a singular matrix A, the expected number of incorrect classifications is one.

One should also be fully aware of the underlying statistical bias of the proposed method: it assumes that A is singular, and only if this hypothesis becomes extremely unlikely, i.e., if the *a posteriori* likelihood is smaller than ω^ℓ, we switch to the alternative hypothesis that A is nonsingular. In this manner, one can be sure that if A is singular, the probability that the proposed method claims that it is nonsingular is bounded by ω^ℓ, a level of uncertainty under control of the user. If the algorithm does not reject the initial hypothesis, this does not imply that A is singular, and, in fact, the proposed algorithm will simply claim that it is undecided. *The proposed method can only reliably detect nonsingularity*.

It is well-known that the CG algorithm usually does not converge within n steps for an $n \times n$ matrix in inexact arithmetic (40). It is thus necessary to include a stopping criterion. We propose to stop the CG algorithm at step m if the computation errors have become greater than the norm of the current approx-

imation, i.e., if $\chi^{(m)} \geqslant \eta^{(m)}$; or, if the upper bound on the components in the null space is sufficiently small, i.e., if $\eta^{(m)} + \chi^{(m)} \leqslant \omega$. Note that the latter criterion is valid, because Lemma 2.3 applies for all m; thus, the most restrictive bound applies, and we may stop if we encounter a bound that is sufficiently restrictive for our purpose.

3. Testing nondefectivity of Segre varieties

Algorithm S is suited for testing whether secant varieties of certain algebraic varieties, such as Segre, Segre–Veronese, Grassmann, and subspace varieties, admit the maximum dimension. In this context, one is interested in the mathematically exact rank of the matrix representing the tangent space, the parameters of which can be represented exactly. We opted to investigate Segre varieties because they are understood relatively well, with several proven results about defectivity and nondefectivity, providing ample opportunities for verifying the correctness of the proposed algorithm.

In this paper, we focus on proving subabundance for r and superabundance for R of a Segre variety $\mathscr{S} = \mathbb{PC}^{n_1} \times \mathbb{PC}^{n_2} \times \cdots \times \mathbb{PC}^{n_d}$, where r and R are as in (1.5). Recall that this is sufficient for proving generic nondefectivity of $\mathscr{S}$ by Remark 3.3 in (4). The algorithm we propose is essentially the classic algorithm based on Terracini's lemma that we discussed following Theorem 1.2, with two major alterations: we replace the rank computation with a test for singularity using Algorithm S, and combine the tests for sub- and superabundance following Theorem 3.1 that will be introduced here.

We summarize the various matrices related to (parts of) tangent spaces of Segre varieties that will be used throughout this section:

$$\begin{aligned} T_i^k &= \mathbf{a}_i^1 \otimes \cdots \otimes \mathbf{a}_i^{k-1} \otimes I_{n_k} \otimes \mathbf{a}_i^{k+1} \otimes \cdots \otimes \mathbf{a}_i^d, \\ T_i &= \begin{bmatrix} T_i^1 & T_i^2 & \cdots & T_i^d \end{bmatrix}, \\ T^k &= \begin{bmatrix} T_1^k & T_2^k & \cdots & T_r^k \end{bmatrix}, \text{ and} \\ \mathscr{T} &= \begin{bmatrix} T^1 & T^2 & \cdots & T^d \end{bmatrix}. \end{aligned}$$

The core of the proposed method consists of constructing the tangent space matrix $\mathscr{T}$ in (1.4) and verifying that it is of the expected rank. For applying Algorithm S, however, we must first construct a new matrix that is of full rank if and only if $\mathscr{T}$ is of full rank. We note that the parametrization of the tangent space at a smooth[7] point $p_i \in \mathscr{S}$, i.e.,

$$T_i := \begin{bmatrix} T_i^1 & T_i^2 & \cdots & T_i^d \end{bmatrix} \in \mathbb{R}^{\Pi \times \Sigma}, \text{ with } T_i^k \text{ as in (1.3)},$$

is not of full column rank because the dimension of the Segre variety $\mathscr{S}$ is at most $\Sigma - d$. Therefore, we may select $\Sigma - d + 1$ columns from T_i still spanning its range by constructing a $\Sigma \times (\Sigma - d + 1)$ permutation matrix[8] P_i so that $\operatorname{range}(T_i) = \operatorname{range}(T_i P_i)$. First note that

$$\mathbf{a}_i^1 \otimes \cdots \otimes \mathbf{a}_i^d = T_i^1 \mathbf{a}_i^1 = T_i^k \mathbf{a}_i^k, \quad \text{for all } k \geqslant 2;$$

that is, the combined span of T_i^1 and T_i^k always contains at least one redundant column. Letting j be such that the jth component of $\mathbf{a}_i^k$ is nonzero, we may consider P_i^k to be the I_{n_k} identity matrix whose

[7] A smooth point of the Segre variety $\mathscr{S}$ is a point where the local dimension is maximal, i.e., where the tangent space matrix T_i is of maximal rank $\Sigma - d + 1$. It is assumed in the remainder that all points are smooth. Note that this assumption is not crucial; if one of the selected points is not smooth, the algorithm would detect that the matrix is singular.

[8] A permutation matrix is a binary matrix in which every column contains exactly one 1 and every row contains at most one 1.

jth column has been removed. Then, the span of $\begin{bmatrix} T_i^1 & T_i^k \end{bmatrix}$ coincides with the span of $\begin{bmatrix} T_i^1 & T_i^k P_i^k \end{bmatrix}$, as applying P_i^k on the right of T_i^k simply removes the jth column that is linearly dependent on the remaining columns. It follows[9] that the column span of T_i coincides with

$$\begin{bmatrix} T_i^1 & T_i^2 P_i^2 & \cdots & T_i^d P_i^d \end{bmatrix} = T_i \cdot \text{block-diag}\left(I_{n_1}, P_i^2, P_i^3, \ldots, P_i^d\right) := T_i P_i,$$

a matrix that has precisely $\Sigma - d + 1$ columns, so that it is of full rank iff $p_i \in \mathscr{S}$ is smooth. Herein, block-diag$(\cdot)$ is a block diagonal matrix with the given matrices on the block diagonal. Next, we note that the span of (1.4), i.e., $\mathscr{T} = \begin{bmatrix} T_1^1 & \cdots & T_r^1 & T_1^2 & \cdots & T_r^2 & \cdots & T_1^d & \cdots & T_r^d \end{bmatrix}$, coincides with the column span of

$$\begin{bmatrix} T_1^1 & \cdots & T_r^1 & T_1^2 P_1^2 & \cdots & T_r^2 P_r^2 & \cdots & T_1^d P_1^d & \cdots & T_r^d P_r^d \end{bmatrix} = \mathscr{T} P,$$

wherein one may write

$$\begin{aligned} P &= \text{block-diag}\left(I_{rn_1}, \text{block-diag}\left(P_1^2, \ldots, P_r^2\right), \ldots, \text{block-diag}\left(P_1^d, \ldots, P_r^d\right)\right) \\ &:= \text{block-diag}\left(I_{rn_1}, P^2, P^3, \ldots, P^d\right) \\ &:= \text{block-diag}\left(I_{rn_1}, P_\star\right), \end{aligned}$$

with $P_\star := \text{block-diag}\left(P^2, P^3, \ldots, P^d\right)$, so that $\text{range}\left(\mathscr{T}\right) = \text{range}\left(\mathscr{T} P\right)$. Remark that P is an $r\Sigma \times r(\Sigma - d + 1)$ matrix; thus, if $\mathscr{T} P$ is of full column rank, then the rth order secant variety is nondefective.[10] Our second key observation is that subabundance and superabundance may be verified simultaneously:

THEOREM 3.1 Let $c := \Pi - r(\Sigma - d + 1)$ and T_R be the parameterization of the tangent space of $p_R \in \mathscr{S}$. If there exists a matrix $C_R \in \mathbb{R}^{\Sigma - d + 1 \times c}$ of full column rank such that

$$\text{rank}\left(\begin{bmatrix} \mathscr{T} P & T_R P_R C_R \end{bmatrix}\right) = \text{rank}\left(\begin{bmatrix} \mathscr{T} & T_R \end{bmatrix}\right) = \Pi, \tag{3.1}$$

then the Segre variety $\mathscr{S}$ is nondefective, i.e., the rth order secant variety is subabundant and the Rth order secant variety is superabundant.

Conversely, if $\mathscr{S}$ is nondefective and the points $p_i \in \mathscr{S}$, $i = 1, \ldots, R$ are generic, then a matrix $C_R \in \mathbb{R}^{\Sigma - d + 1 \times c}$ whose entries are drawn i.i.d. from the standard normal distribution $N(0,1)$ satisfies (3.1) with probability one.

Proof. One verifies that $0 \leqslant c < \Sigma - d + 1$. Assume there exists a matrix C_R such that (3.1) holds. From

$$\Pi = \text{rank}\left(\begin{bmatrix} \mathscr{T} P & T_R P_R C_R \end{bmatrix}\right) \leqslant \text{rank}\left(\mathscr{T} P\right) + \text{rank}\left(T_R P_R C_R\right) \leqslant r(\Sigma - d + 1) + \Pi - r(\Sigma - d + 1) = \Pi,$$

it is clear that the rth order secant variety is subabundant because $\mathscr{T} P$ has to be of full rank for the left equality to hold. Because $\text{range}\left(T_R P_R C_R\right) \subseteq \text{range}\left(T_R\right)$ and the left equality holds, we know that $\begin{bmatrix} \mathscr{T} P & T_R \end{bmatrix}$ is also of full rank, so that the Rth order secant variety is superabundant. This proves the first part of the theorem.

Conversely, as the points on the Segre variety are generic, and the variety is nondefective, the last equality in (3.1) holds vacuously. Let Q_1 be an orthogonal basis for $\text{range}\left(\mathscr{T}\right) = \text{range}\left(\mathscr{T} P\right)$, and let Q_2 be a matrix with orthonormal columns such that $\begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$ is an orthogonal basis of $\mathbb{R}^\Pi$. Then,

$$T_R P_R = Q_1 Q_1^T T_R P_R + Q_2 Q_2^T T_R P_R := Q_1 Q_1^T T_R P_R + Q_2 B,$$

[9] Simply consider the sum of the vector spaces represented by the column span of these matrices and note that the vector space corresponding to T_i^1 appears in each of these terms, so that all repetitions may obviously be removed from the considered sum.

[10] The converse is not necessarily true, because the selected points on $\mathscr{S}$ are not known to be generic.

where $B = Q_2^T T_R P_R$. By assumption, B is of full rank; otherwise, adding p_R would not cause $\mathscr{S}$ to be superabundant, which is contradictory. Thus, the singular value decomposition of B may be written as $B = USV^T$ where $U \in \mathbb{R}^{c\times c}$, $S \in \mathbb{R}^{c\times c}$, and $V \in \mathbb{R}^{\Sigma-d+1\times c}$. Next, we note that both $Q_2' := Q_2 US$ and $C_R' := V^T C_R$ are, with probability one, matrices of rank c: Q_2' is a matrix with c orthogonal columns; and $C_R' \in \mathbb{R}^{c\times c}$ is an orthogonal transformation of Gaussian i.i.d. random variables, which is again standard normally distributed (20, §4.2), i.e., $(C_R')_{ij} \sim N(0,1)$. Consequently, applying the transformation C_R on the right of $(I - Q_1 Q_1^T)T_R P_R = Q_2 Q_2^T T_R P_R$ does not decrease its rank; hence,

$$\begin{aligned}\Pi = \operatorname{rank}\left(\begin{bmatrix}\mathscr{T} & T_R\end{bmatrix}\right) &= \operatorname{rank}(\mathscr{T}) + \operatorname{rank}\left((I - Q_1 Q_1^T)T_R\right)\\ &= \operatorname{rank}(\mathscr{T}P) + \operatorname{rank}\left((I - Q_1 Q_1^T)T_R P_R C_R\right)\\ &= \operatorname{rank}\left(\begin{bmatrix}\mathscr{T}P & T_R P_R C_R\end{bmatrix}\right),\end{aligned}$$

with probability one, completing the proof. □

The above theorem halves the required number of rank computations in the classic algorithm based on Terracini's lemma. However, Algorithm S cannot be applied directly to this matrix because it is not spsd. To solve this issue, we consider the matrix

$$Z = \begin{bmatrix}T^1 & T_R & T^2 & T^3 & \cdots & T^d\end{bmatrix} \cdot \text{block-diag}\left(I_{rn_1}, P_R C_R, P^2, P^3, \ldots, P^d\right),$$

where T^k is as in (1.4). The Gram matrix of Z is

$$G = Z^T Z = \begin{bmatrix}D & L^T\\ L & S\end{bmatrix}, \quad \text{where} \quad D = \begin{bmatrix}D_1 & L_{1,2}^T\\ L_{1,2} & D_2\end{bmatrix} \quad \text{and} \quad L = \begin{bmatrix}L_1 & L_2\end{bmatrix}, \tag{3.2}$$

where the matrices on the diagonal are given by

$$D_1 = (T^1)^T T^1,\ D_2 = C_R^T P_R^T T_R^T T_R P_R C_R,\ S = P_\star^T S^\star P_\star = P_\star^T \begin{bmatrix}(T^2)^T T^2 & (T^2)^T T^3 & \cdots & (T^2)^T T^d\\ (T^3)^T T^2 & (T^3)^T T^3 & \cdots & (T^3)^T T^d\\ \vdots & \vdots & \ddots & \vdots\\ (T^d)^T T^2 & (T^d)^T T^3 & \cdots & (T^d)^T T^d\end{bmatrix} P_\star,$$

and the off-diagonal matrices are: $L_{1,2} := C_R^T P_R^T L_{1,2}^\star = C_R^T P_R^T T_R^T T^1$,

$$L_1 := P_\star^T L_1^\star = P_\star^T \begin{bmatrix}(T^2)^T T^1\\ (T^3)^T T^1\\ \vdots\\ (T^d)^T T^1\end{bmatrix}, \quad \text{and} \quad L_2 := P_\star^T L_2^\star P_R C_R = P_\star^T \begin{bmatrix}(T^2)^T T_R\\ (T^3)^T T_R\\ \vdots\\ (T^d)^T T_R\end{bmatrix} P_R C_R.$$

We may then apply the Guttman–Haynsworth rank additivity formula (26; 27) twice to the Gram matrix G:[11]

$$\operatorname{rank}\left(\begin{bmatrix}\mathscr{T} & T_R\end{bmatrix}\right) = \operatorname{rank}(Z) = \operatorname{rank}(G) = \operatorname{rank}(D_1) + \operatorname{rank}(D/D_1) + \operatorname{rank}(G/D),$$

where the Schur complements are defined by

$$D/D_1 := D_2 - L_{1,2} D_1^{-1} L_{1,2}^T \quad \text{and} \quad G/D := S - L D^{-1} L^T.$$

[11] In principle, the first equality holds only with probability one. With probability zero, the rank of Z may be strictly less than that of the matrix representation of the tangent space; this will not alter the probability of failure, however.

Applying the inversion formula of a partitioned block matrix in **(author?)** (26), one readily finds:

$$G/D = S - L_1 D_1^{-1} L_1^T - (L_2 - L_1 D_1^{-1} L_{1,2}^T)(D/D_1)^{-1}(L_2 - L_1 D_1^{-1} L_{1,2}^T)^T. \tag{3.3}$$

As inertia is additive on the Schur complement (27), the eigenvalues of G, D_1, D/D_1, and G/D are all nonnegative and real. We thus obtain

COROLLARY 3.1 If G/D, D/D_1 and D_1 are all nonsingular, then the Segre variety $\mathscr{S}$ is nondefective.

Algorithm S can be applied to each of these three spsd matrices. In Corollary 3.2, it will be revealed that testing nonsingularity of D_1 and D/D_1 is not necessary in our implementation, so we do not test them explicitly.

ALGORITHM D. The above considerations lead us to propose the following algorithm for verifying whether a Segre variety $\mathscr{S}$ is not defective. Parameterize R points on $\mathscr{S}$ by generating vectors $\{\mathbf{a}_i^k \in \mathbb{R}^{n_k}\}_{i=1}^R$, $k = 1,\ldots,d$, whose components are i.i.d. standard normally distributed, so that the tensors $\mathbf{a}_i^1 \otimes \cdots \otimes \mathbf{a}_i^d$, $i = 1,\ldots,R$, represent exact mathematical points on $\mathscr{S}$. We will verify whether the Segre variety $\mathscr{S}$ is nondefective by applying Terracini's lemma to these points, (implicitly) constructing the tangent space matrices $\mathscr{T}$ and T_R, and then checking whether the rank of G/D is maximal by applying Algorithm S to it with input parameters ω and ℓ. If Algorithm S claims that G/D is nonsingular, the proposed algorithm claims that the Segre variety is nondefective. On the other hand, if Algorithm S claims that it is undecided, the proposed algorithm also claims that it is undecided. The probability that the proposed algorithm incorrectly claims that a defective variety is nondefective is bounded by ω^ℓ.

3.1 *A structure-exploiting matrix-vector product*

Applying Algorithm S during the course of Algorithm D requires an implementation of a matrix-vector product with G/D as well as a practical bound on the forward error $\delta(G/D,\mathbf{x})$ of this operation. Here, we propose a memory efficient structure-exploiting implementation *that avoids constructing G/D explicitly*; in the next subsection, a numerical analysis of this implementation is presented.

Consider the structure of $\mathscr{T}^T\mathscr{T}$; it admits a block structure inherited from (1.3):

$$\mathscr{T}^T\mathscr{T} = \begin{bmatrix} D_1 & L_1^{\star T} \\ L_1^\star & S^\star \end{bmatrix} = \begin{bmatrix} (T^1)^T T^1 & (T^1)^T T^2 & \cdots & (T^1)^T T^d \\ (T^2)^T T^1 & (T^2)^T T^2 & \cdots & (T^2)^T T^d \\ \vdots & \vdots & \ddots & \vdots \\ (T^d)^T T^1 & (T^d)^T T^2 & \cdots & (T^d)^T T^d \end{bmatrix}; \tag{3.4}$$

we additionally note that the blocks $(T^i)^T T^j$ admit a block structure inherited from (1.4):

$$(T^i)^T T^j = \begin{bmatrix} (T_1^i)^T T_1^j & (T_1^i)^T T_2^j & \cdots & (T_1^i)^T T_r^j \\ (T_2^i)^T T_1^j & (T_2^i)^T T_2^j & \cdots & (T_2^i)^T T_r^j \\ \vdots & \vdots & \ddots & \vdots \\ (T_r^i)^T T_1^j & (T_r^i)^T T_2^j & \cdots & (T_r^i)^T T_r^j \end{bmatrix}. \tag{3.5}$$

From (1.3) one derives that the blocks in the above matrix are given by

$$(T_g^i)^T T_h^i = \prod_{\substack{k=1 \\ k\neq i}}^d \langle \mathbf{a}_g^k, \mathbf{a}_h^k \rangle \; I_{n_i}, \qquad \text{and, if } i \neq j, \qquad (T_g^i)^T T_h^j = \prod_{\substack{k=1 \\ k\neq i,j}}^d \langle \mathbf{a}_g^k, \mathbf{a}_h^k \rangle \; \mathbf{a}_h^i (\mathbf{a}_g^j)^T; \tag{3.6}$$

i.e., if $i = j$, the (g,h) block of $(T^i)^T T^j$ is a scaled diagonal matrix, and otherwise it is a scaled rank-1 matrix. Letting

$$A^{(k)} = \begin{bmatrix} \mathbf{a}_1^k & \mathbf{a}_2^k & \cdots & \mathbf{a}_r^k \end{bmatrix} \quad \text{and} \quad J^{(k)} := (A^{(k)})^T A^{(k)},$$

it follows from (3.5) and (3.6) that

$$(T^i)^T T^i = \bigodot_{\substack{k=1 \\ k \neq i}}^{d} \left((A^{(k)})^T A^{(k)} \right) \otimes I_{n_i} = \bigodot_{\substack{k=1 \\ k \neq i}}^{d} J^{(k)} \otimes I_{n_i} := H^{(i)} \otimes I_{n_i}, \tag{3.7}$$

where $A \odot B$ denotes the Hadamard product of the matrices A and B. Because the inner product matrices $J^{(k)}$ are often required in Algorithm 2, we compute them beforehand and provide them as input. For future reference, define the following block partitioning of the vectors $\mathbf{y}$ and $\mathbf{z}$ that is compatible with the two-level block structure (3.4)–(3.5) for the operation $\mathbf{y} = \mathscr{T}^T \mathscr{T} \mathbf{z}$:

$$\mathbf{y}^T := \begin{bmatrix} \mathbf{y}_1^T & \mathbf{y}_\star^T \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1^T & \mathbf{y}_2^T & \cdots & \mathbf{y}_d^T \end{bmatrix}, \quad \text{and} \quad \mathbf{z}^T := \begin{bmatrix} \mathbf{z}_1^T & \mathbf{z}_\star^T \end{bmatrix} = \begin{bmatrix} \mathbf{z}_1^T & \mathbf{z}_2^T & \cdots & \mathbf{z}_d^T \end{bmatrix}, \tag{3.8}$$

with $\mathbf{y}_i, \mathbf{z}_i \in \mathbb{R}^{n_i r}$, and let every $\mathbf{y}_i$ and $\mathbf{z}_i$ be partitioned as

$$\mathbf{y}_i^T := \begin{bmatrix} \mathbf{y}_{i,1}^T & \mathbf{y}_{i,2}^T & \cdots & \mathbf{y}_{i,r}^T \end{bmatrix} \quad \text{and} \quad \mathbf{z}_i^T := \begin{bmatrix} \mathbf{z}_{i,1}^T & \mathbf{z}_{i,2}^T & \cdots & \mathbf{z}_{i,r}^T \end{bmatrix}, \tag{3.9}$$

where $\mathbf{y}_{i,j}, \mathbf{z}_{i,j} \in \mathbb{R}^{n_i}$. Let us define furthermore the matrices $Y^{(i)}, Z^{(i)} \in \mathbb{R}^{n_i \times r}$:

$$Y^{(i)} := \begin{bmatrix} \mathbf{y}_{i,1} & \mathbf{y}_{i,2} & \cdots & \mathbf{y}_{i,r} \end{bmatrix} \quad \text{and} \quad Z^{(i)} := \begin{bmatrix} \mathbf{z}_{i,1} & \mathbf{z}_{i,2} & \cdots & \mathbf{z}_{i,r} \end{bmatrix}, \tag{3.10}$$

i.e., $\mathbf{y}_i$ is the vector obtained by stacking the columns of $Y^{(i)}$, and similarly for $\mathbf{z}_i$: $\operatorname{vec}(Y^{(i)}) = \mathbf{y}_i$ and $\operatorname{vec}(Z^{(i)}) = \mathbf{z}_i$. Multiplying (3.7) with a vector on the right can be performed efficiently without computing the Kronecker product; it is namely known from (51) that

$$\mathbf{y}_i = (T^i)^T T^i \mathbf{z}_i = (H^{(i)} \otimes I_{n_i}) \mathbf{z}_i \quad \text{is equivalent with} \quad Y^{(i)} = Z^{(i)} (H^{(i)})^T. \tag{3.11}$$

Similarly, the matrices $(T^i)^T T^j$, $i \neq j$, may also be applied to a vector without computing them:

$$\mathbf{y}_{i,g} = \sum_{h=1}^{r} (T_g^i)^T T_h^j \mathbf{z}_{j,h} = \sum_{h=1}^{r} \Big(\prod_{\substack{k=1 \\ k \neq i,j}}^{d} \langle \mathbf{a}_g^k, \mathbf{a}_h^k \rangle \Big) \mathbf{a}_h^i \langle \mathbf{a}_g^j, \mathbf{z}_{j,h} \rangle = \sum_{h=1}^{r} h_{gh}^{(i,j)} \mathbf{a}_h^i \quad \text{with} \quad h_{gh}^{(i,j)} = \langle \mathbf{a}_g^j, \mathbf{z}_{j,h} \rangle \prod_{\substack{k=1 \\ k \neq i,j}}^{d} \langle \mathbf{a}_g^k, \mathbf{a}_h^k \rangle.$$

This can be rewritten tersely in matrix form: $\mathbf{y}_i := (T^i)^T T^j \mathbf{z}_j$ is equivalent with

$$Y^{(i)} = A^{(i)} (H_{Z^{(j)}}^{(i,j)})^T, \quad \text{where} \quad H_X^{(i,j)} := (A^{(j)})^T X \odot H^{(i,j)} := (A^{(j)})^T X \odot \bigodot_{\substack{k=1 \\ k \neq i,j}}^{d} (A^{(k)})^T A^{(k)}, \tag{3.12}$$

assuming that $X \in \mathbb{R}^{n_j \times r}$. Consider now a matrix-vector product with G/D. We write

$$\widehat{\mathbf{y}} := (G/D)\widehat{\mathbf{z}} := \widehat{\mathbf{y}}^1 - \widehat{\mathbf{y}}^2 := P_\star^T (\mathbf{y}^1 - \mathbf{y}^2),$$

wherein, deriving from (3.3),

$$\widehat{\mathbf{y}}^1 = (S - L_1 D_1^{-1} L_1^T)\widehat{\mathbf{z}} = P_\star^T \left(S^\star - L_1^\star D_1^{-1} L_1^{\star T} \right) P_\star \widehat{\mathbf{z}} \tag{3.13}$$

$$\widehat{\mathbf{y}}^2 = (L_2 - L_1 D_1^{-1} L_{1,2}^T)(D/D_1)^{-1}(L_2 - L_1 D_1^{-1} L_{1,2}^T)^T \widehat{\mathbf{z}}. \tag{3.14}$$

Consider first (3.13). Computing $\mathbf{z}_\star = P_\star \widehat{\mathbf{z}}$, with $\mathbf{z}_\star$ as in (3.8), is straightforward: if the ith row of $P_\star$ has a one in position l_i, i.e., $(p_\star)_{i,l_i} = 1$, then $(z_\star)_i = \widehat{z}_{l_i}$, otherwise $(z_\star)_i = 0$. A similar observation holds for the product with $P_\star^T$. Letting $\dot{\mathbf{y}} := S^\star \mathbf{z}_\star$ where $\dot{\mathbf{y}}$ is partitioned as $\mathbf{y}_\star$ in (3.8), we obtain for $2 \leqslant i \leqslant d$ that

$$\dot{\mathbf{y}}_i := \sum_{j=2}^{d} (T^i)^T T^j \mathbf{z}_j = (T^i)^T T^i \mathbf{z}_i + \sum_{\substack{j=2 \\ j\neq i}}^{d} (T^i)^T T^j \mathbf{z}_j, \quad \text{or,} \quad \dot{Y}^{(i)} := Z^{(i)} H^{(i)} + A^{(i)} \sum_{\substack{j=2 \\ j\neq i}}^{d} (H^{(i,j)}_{Z^{(j)}})^T,$$

where we observed that $H^{(i)}$ is symmetric and applied (3.11) and (3.12), obtaining the right equation. Next, computing $\ddot{\mathbf{y}} := L_1^\star D_1^{-1} (L_1^\star)^T \mathbf{z}_\star$ is investigated. Letting $\dot{\mathbf{w}} := (L_1^\star)^T \mathbf{z}_\star$, we have, by definition,

$$\dot{\mathbf{w}} := \sum_{j=2}^{d} (T^1)^T T^j \mathbf{z}_j; \quad \text{applying (3.12) results in} \quad \dot{W} = A^{(1)} \sum_{j=2}^{d} (H^{(1,j)}_{Z^{(j)}})^T,$$

where $\dot{\mathbf{w}} = \mathrm{vec}(\dot{W})$ in the usual manner. Then, $D_1 = H^{(1)} \otimes I_{n_1}$, and thus $D_1^{-1} = (H^{(1)})^{-1} \otimes I_{n_1}$. Consequently, $\ddot{\mathbf{w}} := D_1^{-1} \dot{\mathbf{w}}$ may be written as

$$\ddot{W} = \dot{W} (H^{(1)})^{-1}, \quad \text{where} \quad \ddot{\mathbf{w}} = \mathrm{vec}(\ddot{W}).$$

This is a consequence of the Kronecker structure of D_1^{-1} (51) and the symmetry of $H^{(1)}$. Assume that $\ddot{\mathbf{y}}$ is partitioned as $\mathbf{y}_\star$ in (3.8), i.e., $\ddot{\mathbf{y}}^T = \begin{bmatrix} \ddot{\mathbf{y}}_2^T & \cdots & \ddot{\mathbf{y}}_d^T \end{bmatrix}$. Then, $\ddot{\mathbf{y}} = L_1^\star \ddot{\mathbf{w}}$ is given as follows:

$$\ddot{\mathbf{y}}_i = (T^i)^T T^1 \ddot{\mathbf{w}}, \quad \text{which is equivalent to} \quad \ddot{Y}^{(i)} = A^{(i)} (H^{(i,1)}_{\ddot{W}})^T,$$

by (3.12). Finally,

$$\mathbf{y}^1 := \dot{\mathbf{y}} - \ddot{\mathbf{y}}, \tag{3.15}$$

so that applying $P_\star^T$ completes the computation of (3.13). The second update term for the Schur complement, presented in (3.14), will be applied through straightforward matrix-vector products because these matrices are sufficiently small to represent explicitly. We compute

$$L_2' := (L_2^\star - L_1^\star D_1^{-1} (L_{1,2}^\star)^T) P_R C_R, \text{ and } D_2' := D/D_1 = C_R^T P_R^T (T_R^T T_R - L_{1,2}^\star D_1^{-1} (L_{1,2}^\star)^T) P_R C_R; \tag{3.16}$$

the correctness of these equations is clear from (3.2) and its succeeding definitions. Then,

$$\mathbf{y}^2 = L_2' ((D_2')^{-1} ((L_2')^T \mathbf{z}_\star)) \tag{3.17}$$

is computable with unstructured matrix-vector products. Applying $P_\star^T$ to it yields (3.14). The above discussion is summarized as Algorithm 2. It is henceforth assumed that Algorithm 2 is used as implementation of the matrix-vector product with G/D, required by Algorithm S in Algorithm D, and whose forward error will be investigated in Theorem 3.2.

We continue with the practical computation of (3.16). Consider the computation of L_2': $L_2^\star$ and $L_{1,2}^\star$ are constructed simply by applying the definitions in (3.6). Consider then $\dot{L} = D_1^{-1} (L_{1,2}^\star)^T$, and let

$$(L_{1,2}^\star)^T := \begin{bmatrix} \mathbf{l}_1^\star & \mathbf{l}_2^\star & \cdots & \mathbf{l}_{\Sigma-d+1}^\star \end{bmatrix} \in \mathbb{R}^{n_1 r \times \Sigma - d + 1} \quad \text{where} \quad \mathrm{vec}(L_{\{i\}}^\star) := \mathbf{l}_i^\star \text{ and } L_{\{i\}}^\star \in \mathbb{R}^{n_1 \times r}.$$

Algorithm 2: Computing a matrix-vector product with the Schur complement G/D.

input : $\{A^{(k)}\}_k$, $\{J^{(k)}\}_k$, L'_2, D'_2, and $\mathbf{z}_\star$ as in (3.8)–(3.10).

output: $\mathbf{y}_\star$ as in (3.8)–(3.10), and $\mathbf{y}_\star = (S^\star - L_1^\star D_1^{-1}(L_1^\star)^T - L'_2(D'_2)^{-1}(L'_2)^T)\mathbf{z}_\star$.

$W^{(1)} \leftarrow 0$
for $j = 2,3,\ldots,d$ **do**
  $K^{(j)} \leftarrow (A^{(j)})^T Z^{(j)}$
  $W^{(1)} \leftarrow W^{(1)} + K^{(j)} \odot \bigodot_{k\neq 1,j} J^{(k)}$
end
$\dot{W} \leftarrow A^{(1)}(W^{(1)})^T$
$H^{(1)} \leftarrow \bigodot_{k\neq 1} J^{(k)}$
$\ddot{W} \leftarrow \dot{W}(H^{(1)})^{-1}$
$C \leftarrow (A^{(1)})^T \ddot{W}$
for $i = 2,3,\ldots,d$ **do**
  $W^{(i)} \leftarrow 0$
  for $j = 2,3,\ldots,d$ **do**
    if $i \neq j$ **then**
      $W^{(i)} \leftarrow W^{(i)} + K^{(j)} \odot \bigodot_{k\neq i,j} J^{(k)}$
    end
  end
  $H^{(i,1)} \leftarrow \bigodot_{k\neq 1,i} J^{(k)}$
  $H^{(i,1)}_{\ddot{W}} \leftarrow C \odot H^{(i,1)}$
  $\ddot{W}^{(i)} \leftarrow W^{(i)} - H^{(i,1)}_{\ddot{W}}$
  $\dot{Y}^{(i)} \leftarrow A^{(i)}(\ddot{W}^{(i)})^T$
  $H^{(i)} \leftarrow J^{(1)} \odot H^{(i,1)}$
  $Y_1^{(i)} \leftarrow \dot{Y}^{(i)} + Z^{(i)} H^{(i)}$
end
$(\mathbf{y}^1)^T \leftarrow \begin{bmatrix} \mathrm{vec}(Y_1^{(2)})^T & \mathrm{vec}(Y_1^{(3)})^T & \cdots & \mathrm{vec}(Y_1^{(d)})^T \end{bmatrix}$
$\mathbf{f}_1 \leftarrow (L'_2)^T \mathbf{z}_\star$
$\mathbf{f}_2 \leftarrow (D'_2)^{-1}\mathbf{f}_1$
$\mathbf{y}^2 \leftarrow L'_2 \mathbf{f}_2$
$\mathbf{y}_\star \leftarrow \mathbf{y}^1 - \mathbf{y}^2$

We may then write the product

$$\dot{\mathbf{l}}_i := D_1^{-1}\mathbf{l}_i^\star \quad \text{equivalently as} \quad \dot{L}_{\{i\}} := L^\star_{\{i\}}(H^{(1)})^{-1},$$

where $\dot{\mathbf{l}}_i = \mathrm{vec}(\dot{L}_{\{i\}})$ in the usual manner. Continuing with $\ddot{L} := L_1^\star \dot{L}$, or, column-wise, $\ddot{\mathbf{l}}_i := L_1^\star \dot{\mathbf{l}}_i$, and partitioning $\ddot{\mathbf{l}}_i$ like $\mathbf{y}_\star$ in (3.8), we may write, for $2 \leqslant j \leqslant d$,

$$(\ddot{\mathbf{l}}_i)_j := T^{j^T} T^1 \dot{\mathbf{l}}_i, \quad \text{so that} \quad (\ddot{\mathbf{l}}_i)_j = \mathrm{vec}(A^{(j)} H^{(j,1)}_{\dot{L}_{\{i\}}}).$$

by definition (3.12). Finally, having computed $L_2^\star$ and $\ddot{L}$ explicitly, we can compute $L'_2 := (L_2^\star - \ddot{L})P_R C_R$ using unstructured operations. Computing D'_2 is accomplished as follows: $T_R^T T_R$ can be constructed

explicitly by considering the definitions in (3.6). The product $\dot{L} = D_1^{-1}(L_{1,2}^{\star})^T$ was already detailed and $L_{1,2}^{\star}$ is also available explicitly, so that we can compute $D_2' = C_R^T P_R^T ((T_R^T T_R - L_{1,2}^{\star}\dot{L}) P_R C_R)$ using unstructured operations. In the appendix, we present Algorithm 3 formalizing the above discussion; its numerical properties are investigated in Lemma 3.3.

3.2 *Numerical analysis*

Having introduced the implementation of the matrix-vector product with G/D in Algorithm 2, we continue by deriving a computable upper bound on the forward error $\delta(G/D, \mathbf{x})$. This bound may then be combined with Lemma 2.3 for practically computing $\chi^{(m)}$, required by Algorithm S, and, hence, Algorithm D. We should first settle our notation and recall some basic identities from (31; 53) which will be used in further derivations. First,

$$\prod_{i=1}^{k}(1+\delta_i) = 1+\theta_k, \quad \text{where} \quad |\theta_k| \leqslant \varepsilon_k := \frac{k\varepsilon}{1-k\varepsilon},$$

provided that $|\delta_i| \leqslant \varepsilon$ and $1 - k\varepsilon > 0$. From this, one can deduce that

$$\left| \mathrm{fl}\left(\sum_{i=1}^{k} a_i\right) - \sum_{i=1}^{k} a_i \right| \leqslant \varepsilon_{k-1} \sum_{i=1}^{k} |a_i|; \qquad \text{and} \qquad \left| \mathrm{fl}\left(\prod_{i=1}^{k} a_i\right) - \prod_{i=1}^{k} a_i \right| \leqslant \varepsilon_{k-1} \prod_{i=1}^{k} |a_i|, \tag{3.18}$$

see, e.g., **(author?)** (53). The forward error in matrix multiplication can also be bounded easily (31): if $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times p}$, then

$$\mathrm{fl}\,(AB) = AB + E \quad \text{with} \quad |E| \leqslant \varepsilon_n |A||B| \leqslant \varepsilon_n \min\{\|A\|_{\max}\|B\|_1, \|A\|_\infty \|B\|_{\max}\}, \tag{3.19}$$

where the absolute value should be taken componentwise. Letting $k_1, k_2 \geqslant 1$, another useful property is

$$\varepsilon_{k_1} + \varepsilon_{k_2} + \varepsilon_{k_1}\varepsilon_{k_2} \leqslant \varepsilon_{k_1+k_2} \quad \text{and, thus,} \quad \varepsilon_{k_1} + \varepsilon_{k_2} \leqslant \varepsilon_{k_1+k_2} \quad \text{and} \quad \varepsilon_{k_1} \leqslant \varepsilon_{k_1+1}, \tag{3.20}$$

assuming all involved terms are smaller than one (31, Lemma 3.3). It is also elementary to prove that $k_2\varepsilon_{k_1} \leqslant \varepsilon_{k_1 k_2}$. Concerning notation, *we abide by the convention that for every matrix* $X \in \mathbb{R}^{m\times n}$

$$\underline{X} = \mathrm{fl}(X) = X + E_X, \quad \text{and, for the sake of compactness,} \quad \underline{X}^{(i)} = \mathrm{fl}(X^{(i)}) = X^{(i)} + E_X^{(i)}.$$

Whenever no explicit reference to a matrix is made, one should assume the corresponding quantity from Section 3.1; for instance, $H^{(i)}$ is as in (3.7), $Y^{(i)}$ as in (3.12), and so on.

Before we can state the main theorem, some preparatory lemmas are introduced; their proofs are included in the appendix. The first lemma concerns the forward error in solving a linear system by multiplying with some approximation of the inverse of the coefficient matrix:

LEMMA 3.1 Let $\underline{X} = X + E_X$, $\underline{Y} = Y + E_Y$ and $\underline{Z} = Z + E_Z$ be matrices of dimensions compatible with $Y \in \mathbb{R}^{n\times n}$, satisfying $Z = XY^{-1}$ and $\underline{Z} = \mathrm{fl}\left(\underline{X}\,\mathrm{fl}\left(\underline{Y}^{-1}\right)\right)$. Let $E_b(\underline{X},\underline{Y},\underline{Z}) = \underline{Z}\,\underline{Y} - \underline{X}$, then,

$$|E_Z| \leqslant \tau\left(\underline{Y}, \mathrm{fl}\left(\underline{Y}^{-1}\right), E_Y\right)\left(\|E_X\|_{\max} + \|E_b(\underline{X},\underline{Y},\underline{Z})\|_{\max} + \|\underline{Z}E_Y\|_{\max}\right),$$

where

$$\tau\left(\underline{Y}, \mathrm{fl}\left(\underline{Y}^{-1}\right), E_Y\right) = \psi^{-1}\sqrt{n}\left\|\mathrm{fl}\left(\underline{Y}^{-1}\right)\right\|_F$$

and

$$\psi := \max\{0, 1 - \left\|I - \mathrm{fl}\left(\underline{Y}^{-1}\right)\underline{Y}\right\|_F - \|\,\mathrm{fl}\left(\underline{Y}^{-1}\right)\|_F \|E_Y\|_F\}.$$

If Y is singular but $\underline{Y}$ is not, then the above bounds are still computable and reduce to $|E_Z| \leqslant +\infty$.

The next lemma characterizes the forward error of computing (3.12) by Algorithm 2:

LEMMA 3.2 Let $\underline{X} = X + E_X$ with $|E_X| \leqslant \varepsilon_X$ and let $\alpha_j := \|A^{(j)}\|_1$. Then,

$$\mathrm{fl}\left(H_{\underline{X}}^{(i,j)}\right) = H_X^{(i,j)} + E_{H_X}^{(i,j)} \quad \text{where} \quad |E_{H_X}^{(i,j)}| \leqslant \varepsilon_{(d-1)(n_1+1)}\alpha_j\|X\|_{\max} + (1+\varepsilon_{(d-1)(n_1+1)})\alpha_j\varepsilon_X.$$

The final preparatory lemma bounds the forward error in computing D_2' and L_2' by Algorithm 3:

LEMMA 3.3 Let $c' := \Sigma - d + 1$, $\alpha^\infty := \max_{2\leqslant j\leqslant d}\|A^{(j)}\|_\infty$, and α_1 be as in Lemma 3.2. Letting $E_b(\cdot,\cdot,\cdot)$ and $\tau(\cdot,\cdot,\cdot)$ be as in Lemma 3.1, define

$$\tau_1 := \tau\left(\underline{H}^{(1)}, \mathrm{fl}((\underline{H}^{(1)})^{-1}), E_H^{(1)}\right)$$

and

$$\mu := 1 + \varepsilon_{(d-1)(n_1+1)}^{-1} \cdot \max_{1\leqslant i\leqslant c'}\|E_b(\underline{L}_{\{i\}}^\star, \underline{H}^{(i)}, \underline{\dot{L}}_{\{i\}})\|_{\max} + \max_{1\leqslant i\leqslant c'}\left\|\underline{\dot{L}}_{\{i\}}\right\|_\infty.$$

Computing $\underline{D}_2' = D_2' + E_{D_2'}$ and $\underline{L}_2' = L_2' + E_{L_2'}$ by Algorithm 3 in finite precision floating-point arithmetic yields

$$|E_{L_2'}| \leqslant (1+\alpha^\infty\alpha_1\tau_1)\varepsilon_{c'+r+1+(\mu+1)(d-1)(n_1+1)}\|C_R\|_1, \quad \text{and}$$
$$|E_{D_2'}| \leqslant \left(\varepsilon_{2c'+1+(d-1)(n_1+1)} + n_1 r\tau_1\varepsilon_{2c'+n_1r+1+(\mu+1)(d-1)(n_1+1)}\right)\|C_R\|_1^2.$$

We may now state the main theorem that we set out to prove in this section:

THEOREM 3.2 Consider all quantities defined in Algorithm 2, and let α_j be as in Lemma 3.2; $\zeta := \sum_{j=1}^d \alpha_j\|Z^{(j)}\|_{\max}$; τ_1 as in Lemma 3.3; and

$$\varepsilon_{\ddot{W}} := \|E_b(\underline{\dot{W}}, \underline{H}^{(1)}, \underline{\ddot{W}})\|_{\max} + \varepsilon_{r+(d-1)(n_1+2)}\zeta\|A^{(1)}\|_\infty + \varepsilon_{(d-1)(n_1+1)}\|\underline{\ddot{W}}\|_\infty,$$

with $E_b(\cdot,\cdot,\cdot)$ defined as in Lemma 3.1. Letting $\tau(\cdot,\cdot,\cdot)$ be as in Lemma 3.1, define

$$\tau_2 := \tau\left(\underline{D}_2', \mathrm{fl}((\underline{D}_2')^{-1}), E_{D_2'}\right).$$

Computing $G/D\mathbf{z}_\star$ by Algorithm 2 in finite precision floating-point arithmetic results in

$$\mathrm{fl}\,(G/D\mathbf{z}_\star) = G/D\mathbf{z}_\star + \mathbf{e} \quad \text{where } \|\mathbf{e}\|_{\max} \leqslant \delta(G/D, \mathbf{z}_\star).$$

The forward error bound satisfies

$$\delta(G/D, \mathbf{z}_\star) = \delta_1 + \delta_2 + \varepsilon(\max_{2\leqslant j\leqslant d}\|\underline{Y}_1^{(j)}\|_{\max} + \|\underline{\mathbf{y}}^2\|_{\max}). \tag{3.21}$$

In (3.21), δ_1 is the forward error attributable to the computation of $\mathbf{y}^1$ in (3.15):

$$\delta_1 := \max_{2\leqslant i\leqslant d}\|E_{Y_1}^{(i)}\|$$

with

$$|E_{Y_1}^{(i)}| \leqslant \|A^{(i)}\|_\infty(\varepsilon_{(d-1)(n_1+3)}(\zeta + \|\underline{C}\|_{\max}) + \varepsilon_{n_1+1}\alpha_1\|\underline{\ddot{W}}\|_{\max} + \alpha_1\tau_1(1+\varepsilon)\varepsilon_{\ddot{W}})$$

$$+\varepsilon_r\|A^{(i)}\|_\infty\|\underline{\ddot{W}}\|_{\max}+\varepsilon_{r+(d-1)(n_1+1)}\|Z^{(i)}\|_\infty+\varepsilon\|\underline{\dot{Y}}^{(i)}\|_{\max}+\varepsilon\|\operatorname{fl}(Z^{(i)}\underline{H}^{(i)})\|_{\max}. \tag{3.22}$$

In (3.21), δ_2 is the forward error in computing $\mathbf{y}^2$ in (3.17):

$$\delta_2=|E_{\mathbf{y}^2}|\leqslant\|E_{L_2'}\|_{\max}\|\underline{\mathbf{f}}_2\|_1+(\|\underline{L}_2'\|_\infty+\|E_{L_2'}\|_\infty)\|E_{\mathbf{f}_2}\|_{\max}+\varepsilon_c\|\underline{L}_2'\|_\infty\|\underline{\mathbf{f}}_2\|_{\max}, \tag{3.23}$$

wherein

$$|E_{\mathbf{f}_2}|\leqslant\tau_2(\|E_{\mathbf{f}_1}\|_{\max}+\|E_b(\underline{\mathbf{f}}_1^T,\underline{D}_2',\underline{\mathbf{f}}_2^T)\|_{\max}+\|E_{D_2'}\|_{\max}\|\underline{\mathbf{f}}_2\|_1);\text{ and} \tag{3.24}$$

$$|E_{\mathbf{f}_1}|\leqslant\varepsilon_{\Pi-n_1r}\|\underline{L}_2'\|_1\|\mathbf{z}_\star\|_{\max}+\|E_{L_2'}\|_{\max}\|\mathbf{z}_\star\|_1, \tag{3.25}$$

and $E_{L_2'}$ and $E_{D_2'}$ may be bounded as in Lemma 3.3.

Proof. The forward error for the computation in line 17 of Algorithm 2 is given by (A.3) in the proof of Lemma 3.2. Analogously, we find

$$\underline{H}^{(i)}=H^{(i)}+E_H^{(i)}\quad\text{where}\quad|E_H^{(i)}|\leqslant\varepsilon_{(d-1)(n_1+1)}.$$

providing the forward error for lines 7 and 21. Note that $|\underline{H}^{(i)}|\leqslant 1$ because even in floating-point arithmetic, the product of numbers smaller than one will remain smaller than one.

The forward error of the computation in lines 1–5 and 11–16 is established by applying Lemma 3.2 with $\underline{X}=\underline{Z}^{(j)}=Z^{(j)}$, so that we may write

$$\underline{H}_{Z^{(j)}}^{(i,j)}=H_{Z^{(j)}}^{(i,j)}+E_{H_{Z^{(j)}}}^{(i,j)}\quad\text{with}\quad|E_{H_{Z^{(j)}}}^{(i,j)}|\leqslant\varepsilon_{(d-1)(n_1+1)}\alpha_j\|Z^{(j)}\|_{\max}.$$

The sum in the loop is handled as follows: using (3.18), we find

$$\Big|\operatorname{fl}\Big(\sum_{j\neq i}\underline{H}_{Z^{(j)}}^{(i,j)}\Big)-\sum_{j\neq i}\underline{H}_{Z^{(j)}}^{(i,j)}\Big|\leqslant\varepsilon_{d-2}\sum_{j\neq i}\Big|\underline{H}_{Z^{(j)}}^{(i,j)}\Big|\leqslant\varepsilon_{d-2}\sum_{j\neq i}\Big(|H_{Z^{(j)}}^{(i,j)}|+|E_{H_{Z^{(j)}}}^{(i,j)}|\Big).$$

Then, the forward error can be bounded as follows:

$$\begin{aligned}|\underline{W}^{(i)}-W^{(i)}|&=\Big|\underline{W}^{(i)}-\sum_{j\neq i}H_{Z^{(j)}}^{(i,j)}-\sum_{j\neq i}E_{H_{Z^{(j)}}}^{(i,j)}+\sum_{j\neq i}E_{H_{Z^{(j)}}}^{(i,j)}\Big|\leqslant\varepsilon_{d-2}\sum_{j\neq i}\Big(\big|H_{Z^{(j)}}^{(i,j)}\big|+\big|E_{H_{Z^{(j)}}}^{(i,j)}\big|\Big)+\sum_{j\neq i}\big|E_{H_{Z^{(j)}}}^{(i,j)}\big|\\&\leqslant\varepsilon_{d-2}\sum_{j\neq i}\alpha_j\|Z^{(j)}\|_{\max}+(1+\varepsilon_{d-2})\varepsilon_{(d-1)(n_1+1)}\sum_{j\neq i}\alpha_j\|Z^{(j)}\|_{\max}\\&\leqslant\varepsilon_{(d-2)+(d-1)(n_1+1)}\sum_{j\neq i}\alpha_j\|Z^{(j)}\|_{\max},\end{aligned}$$

where (A.6) was used in the penultimate step. We obtain $\underline{W}^{(i)}=W^{(i)}+E_W^{(i)}$ with $|E_W^{(i)}|\leqslant\varepsilon_{(d-1)(n_1+2)}\zeta$.

In line 6 we compute

$$\underline{\dot{W}}:=\operatorname{fl}(A^{(1)}(\underline{W}^{(1)})^T)=A^{(1)}(W^{(1)})^T+A^{(1)}(E_W^{(1)})^T+E_{\dot{W}'}^{(1)}:=A^{(1)}(W^{(1)})^T+E_{\dot{W}}^{(1)},$$

where

$$\begin{aligned}|E_{\dot{W}}^{(1)}|&\leqslant|A^{(1)}||E_W^{(1)}|^T+\varepsilon_r|A^{(1)}||W^{(1)}+E_W^{(1)}|^T\leqslant\|A^{(1)}\|_\infty((1+\varepsilon_r)\|E_W^{(1)}\|_{\max}+\varepsilon_r\|W^{(1)}\|_{\max})\\&\leqslant(1+\varepsilon_r)\varepsilon_{(d-1)(n_1+2)}\zeta\|A^{(1)}\|_\infty+\varepsilon_r\zeta\|A^{(1)}\|_\infty\leqslant\varepsilon_{r+(d-1)(n_1+2)}\zeta\|A^{(1)}\|_\infty,\end{aligned}$$

where in the next to last inequality, using (A.6) again, we note that

$$\|W^{(1)}\|_{\max} = \Big\|\sum_{j\neq i} H^{(i,j)}_{Z^{(j)}}\Big\|_{\max} \leqslant \sum_{j\neq i}\Big\|(A^{(j)})^T Z^{(j)}\Big\|_{\max} \leqslant \zeta.$$

The error in the computation on line 8 is handled by applying Lemma 3.1 with $\underline{X} = \dot{\underline{W}}$, $\underline{Y} = \underline{H}^{(1)}$ and $\underline{Z} = \ddot{\underline{W}}$ then we immediately find $\ddot{\underline{W}} = \ddot{W} + E_{\ddot{W}}$ with $|E_{\ddot{W}}| \leqslant \tau_1\varepsilon_{\ddot{W}}$, simply by substituting $\|E^{(1)}_{\dot{W}}\|_{\max}$ and bounding $\|\ddot{\underline{W}}E_{H^{(1)}}\|_{\max} \leqslant \|\ddot{\underline{W}}\|_\infty \|E_{H^{(1)}}\|_{\max}$.

Handling line 18, we first note that

$$\underline{C} = (A^{(1)})^T\ddot{W} + E_C \quad \text{where} \quad |E_C| \leqslant \varepsilon_{n_1}\alpha_1\|\ddot{\underline{W}}\|_{\max} + \alpha_1\|E_{\ddot{W}}\|_{\max}.$$

Then, we essentially apply Lemma 3.2 with a few changes; we have $\underline{H}^{(1,i)}_{\ddot{\underline{W}}} = H^{(1,i)}_{\ddot{W}} + E^{(1,i)}_{H_{\ddot{W}}}$, where, continuing from (A.5),

$$\begin{aligned}
|E^{(1,i)}_{H_{\ddot{W}}}| &\leqslant |\underline{C}| \odot |E^{(1,i)}_{\odot J} + H^{(1,i)}| - H^{(1,i)}_{\ddot{W}} \leqslant |\underline{C}| \odot |E^{(1,i)}_{\odot J}| + |E_C| \odot |H^{(1,i)}| + |E^{(1,i)}_{K\odot J}| \\
&\leqslant (\varepsilon + \varepsilon_{(d-2)(n_1+1)})|\underline{C}| + \varepsilon_{n_1}\alpha_1\|\ddot{\underline{W}}\|_{\max} + \alpha_1\|E_{\ddot{W}}\|_{\max} \\
&\leqslant \varepsilon_{(d-2)(n_1+2)}\|\underline{C}\|_{\max} + \varepsilon_{n_1}\alpha_1\|\ddot{\underline{W}}\|_{\max} + \alpha_1\tau_1\varepsilon_{\ddot{W}},
\end{aligned}$$

where we noted in the third step that $|E^{(1,i)}_{K\odot J}| \leqslant \varepsilon|\underline{C}||\underline{H}^{(1,i)}| \leqslant \varepsilon|\underline{C}|$, because the product of floating-point numbers smaller than 1 in absolute value remains smaller than 1.

The forward error in line 19 satisfies $\ddot{\underline{W}}^{(i)} = \ddot{W}^{(i)} + E^{(i)}_{\ddot{W}}$, with

$$\begin{aligned}
|E^{(i)}_{\ddot{W}}| &\leqslant (1+\varepsilon)\left(|E^{(i)}_W| + |E^{(1,i)}_{H_{\dot{W}}}|\right) + \varepsilon|W^{(i)}| + \varepsilon|H^{(1,i)}_{\dot{W}}| \\
&\leqslant (1+\varepsilon)\left(\varepsilon_{(d-1)(n_1+2)}\zeta + \varepsilon_{(d-2)(n_1+2)}\|\underline{C}\|_{\max} + \varepsilon_{n_1}\alpha_1\|\ddot{\underline{W}}\|_{\max} + \alpha_1\tau_1\varepsilon_{\ddot{W}}\right) + \varepsilon\zeta + \varepsilon\|\underline{C}\|_{\max} \\
&\leqslant \varepsilon_{(d-1)(n_1+3)}(\zeta + \|\underline{C}\|_{\max}) + \varepsilon_{n_1+1}\alpha_1\|\ddot{\underline{W}}\|_{\max} + \alpha_1\tau_1(1+\varepsilon)\varepsilon_{\ddot{W}}.
\end{aligned}$$

Line 20 computes $\dddot{\underline{Y}}^{(i)} := \mathrm{fl}(A^{(i)}\ddot{\underline{W}}^{(i)}) = A^{(i)}\ddot{W}^{(i)} + A^{(i)}E^{(i)}_{\ddot{W}} + E^{(i)}_{\dot{Y}'} := A^{(i)}\ddot{W}^{(i)} + E^{(i)}_{\dot{Y}}$, where the error is readily bounded by

$$|E^{(i)}_{\dot{Y}}| \leqslant \|A^{(i)}\|_\infty\|E^{(i)}_{\ddot{W}}\|_{\max} + \varepsilon_r\|A^{(i)}\|_\infty\|\ddot{\underline{W}}\|_{\max}. \tag{3.26}$$

In line 22 we compute $\ddot{\underline{Y}}^{(i)} := \mathrm{fl}(Z^{(i)}\underline{H}^{(i)}) = Z^{(i)}H^{(i)} + Z^{(i)}E^{(i)}_H + E^{(i)}_{ZH'} := Z^{(i)}H^{(i)} + E^{(i)}_{ZH}$, where

$$|E^{(i)}_{ZH}| \leqslant |Z^{(i)}||E^{(i)}_H| + \varepsilon_r|Z^{(i)}||\underline{H}^{(i)}| \leqslant \varepsilon_{r+(d-1)(n_1+1)}\|Z^{(i)}\|_\infty, \tag{3.27}$$

so that $\underline{Y}^{(i)}_1 := \mathrm{fl}(\dot{\underline{Y}}^{(i)} + \ddot{\underline{Y}}^{(i)}) = Y^{(i)}_1 + E^{(i)}_{\dot{Y}} + E^{(i)}_{ZH} + E^{(i)}_{Y'_1} := Y^{(i)}_1 + E^{(i)}_{Y_1}$ with $|E^{(i)}_{Y'_1}| \leqslant \varepsilon|\dot{\underline{Y}}^{(i)}| + \varepsilon|\,\mathrm{fl}(Z^{(i)}\underline{H}^{(i)})|$. Combining the latter contribution to $|E^{(i)}_{Y_1}|$ with (3.26) and (3.27), we obtain (3.22).

We conclude the analysis with lines 25–28. First, we compute

$$\underline{\mathbf{f}}_1 := \mathrm{fl}\left((\underline{L}'_2)^T\mathbf{z}_\star\right) = L'^T_2\mathbf{z}_\star + E^T_{L'_2}\mathbf{z}_\star + E_{\mathbf{f}'_1} := \mathbf{f}_1 + E_{\mathbf{f}_1},$$

where the error term is readily bounded by (3.25). The error of the linear system solve is obtained by noting that Lemma 3.1 may be applied to the transpose of $\underline{D}'_2\underline{\mathbf{f}}_2 = \underline{\mathbf{f}}_1$, exploiting that $\underline{D}'_2$ is symmetric.

Applying the lemma, we immediately find (3.24). Note that an upper bound on $\|E_{D_2'}\|_{\max}$ is given by Lemma 3.3. The final matrix product is

$$\underline{\mathbf{y}}^2 := \mathrm{fl}\left(\underline{L_2'}\underline{\mathbf{f}}_2\right) = L_2'\mathbf{f}_2 + E_{L_2'}\underline{\mathbf{f}}_2 + L_2'E_{\mathbf{f}_2} + E'_{\mathbf{y}^2} := \mathbf{y}^2 + E_{\mathbf{y}^2},$$

so that the error term is bounded by (3.23). The bound for $\|E_{L_2'}\|_{\max}$ is given by Lemma 3.3. This concludes the proof. □

We mentioned that, in the considered numerical setting, it is not required to compute the rank of $D_1 = H^{(1)} \otimes I$ and $D_2' = D/D_1$. We state it formally as

COROLLARY 3.2 Algorithm D deterministically claims that it is undecided if D_1 or D/D_1 is singular.

Proof. From Lemma 3.1 it follows that if D_1 or D/D_1 is singular, then the forward error of a linear solve with these matrices is bounded by ∞. By Lemma 3.3 and Theorem 3.2, this bound is propagated, so that in either case $\delta(G/D, \mathbf{z}_\star) \leqslant +\infty$ is obtained, causing Algorithm S to proclaim that it is undecided. □

This lemma does not alter the upper bound on the probability that Algorithm D wrongly proclaims that a defective variety is nondefective: it is still bounded by ω^ℓ. In the next section, we present an example of the behavior of the proposed algorithm when D_2' can be considered numerically singular.

Lemma 3.1 is stated independently of the algorithm computing $\mathrm{fl}\left(\underline{Y}^{-1}\right)$. In our implementation, we compute this inverse by inverting an LU decomposition of $\underline{Y}$. Should this factorization fail because $\underline{Y}$ is exactly singular, which due to roundoff errors is extremely unlikely, we halt Algorithm D and let it proclaim that it is undecided.

3.3 *Complexity analysis*

Algorithm D has two advantages over computing the rank of the tangent space matrix using the classic algorithm outlined in the introduction.

Ignoring the $\mathcal{O}(r^3 + (\Sigma + d)r^2)$ setup cost to compute the $J^{(k)}$ and $(H^{(1)})^{-1}$, and the $\mathcal{O}(\Sigma\Pi)$ cost to compute D_2' and L_2' with Algorithm 3, the cost of applying a matrix-vector product with G/D is given as follows. The loop in lines 2–5 in Algorithm 2 costs a total $\mathcal{O}(dr^2 + r\Sigma)$ operations. Lines 6–9 require $\mathcal{O}(n_1 r^2)$ operations, assuming that $(H^{(1)})^{-1}$ is precomputed. The inner loop in lines 12–16 costs a total of $\mathcal{O}(d^3 r^2)$ operations over all iterations of the outer loop. Similarly, over all iterations of the outer loop, lines 17–22 require $\mathcal{O}((d^2 + \Sigma - n_1)r^2)$ operations. Combined with the $\mathcal{O}(\Sigma\Pi)$ operations in line 25, the total cost for computing $\mathbf{y} = G/D\mathbf{z}$ amounts to

$$\mathcal{O}(\Sigma\Pi + dr^2 + r\Sigma + d^3r^2 + (d^2+\Sigma)r^2) = \mathcal{O}\left(\left(\frac{d^3}{\Sigma^2} + \frac{1}{\Sigma}\right)\Pi^2\right)$$

operations. As the number of iterations for convergence in the CG method is $\mathcal{O}(\Pi/n_1)$, it follows that iteratively solving ℓ systems with G/D to assess nonsingularity will usually require less operations than the classic algorithm (provided that Σ is sufficiently large w.r.t. d), whose cost is $\mathcal{O}(\Pi^3)$.

The key advantage of Algorithm D is its reduced memory cost w.r.t. the classic algorithm. The latter stores $\mathcal{O}(\Pi^2)$ values to represent the tangent space matrix; the memory cost of the former is established as follows. We store dr^2 values to represent $\{J^{(k)}\}_k$, Σr values to represent $\{A^{(k)}\}_k$, $\mathcal{O}(\Sigma^2)$ values for D_2', and $\mathcal{O}(r\Sigma(\Sigma - d + 1 - n_1))$ values for L_2'. We also store three vectors of length $r(\Sigma - n_1)$ during the execution of the CG method. One may verify that none of the operations in Algorithm 2 requires more memory than the aforementioned storage costs; the asymptotic memory consumption thus remains

unaffected. We may conclude that Algorithm D stores

$$\mathcal{O}(r\Sigma^2 + dr^2) = \mathcal{O}\left(\Sigma\Pi + \frac{d}{\Sigma^2}\Pi^2\right)$$

values, which is significantly less than the classic algorithm. In Section 4.3, a comparison between the classic and proposed algorithm in terms of memory cost and execution time is presented.

4. Experimental results

In this section, we discuss the details of our computer implementation of Algorithm D. Its typical behavior w.r.t. Lemma 2.3 is investigated in greater detail. Thereafter, an indicative comparison is made with the classic algorithm in terms of execution time and memory consumption. Finally, we present original results pertaining to the nondefectivity of Segre varieties obtained with Algorithm D.

4.1 *Implementation details*

Algorithm D was implemented in C++ in finite precision floating-point arithmetic. Our code employs the Eigen library (25), which implements basic data structures and key algorithms such as matrix multiplication. Eigen allows the user to extend the library with other numerical data types; we extended it with finite precision floating-point data types provided by the QD library (29; 30): the `dd_real` class offers double–double floating-point arithmetic satisfying the standard model of floating-point arithmetic in (2.1) with $\varepsilon = 2^{-106} \approx 1.23 \cdot 10^{-32}$ and `qd_real` offers quadruple–double floating-point arithmetic with $\varepsilon = 2^{-212} \approx 1.52 \cdot 10^{-64}$. Most of our experiments were performed with `dd_real` because its arithmetic is much faster and consumes only half of the memory required by `qd_real`; i.e., 16 versus 32 bytes of memory.

Algorithm D was compared with the classic algorithm implemented in Maple 16 using *exact arithmetic* by generating matrices whose integer components are drawn from a uniform distribution between -100 and 100, forming the tangent space matrix exactly, and then computing its exact rank with the `Rank` function. Comparisons with this algorithm are only indicative of true performance, as the computational kernels of Maple might be implemented more efficiently. In addition, a numerical implementation of the classic algorithm was developed in C++ using Eigen and QD: we compute a rank-revealing QR decomposition using the Eigen method `fullPivHouseholderQr` in double–double precision and determine largest offset k such that the norm of the matrix consisting of the last $\Pi - k$ rows is smaller in magnitude than 10^{-16}; the numerical rank is then $\Pi - k$.[12] To our knowledge, this algorithm has not been modified so as to make it fully reliable in a numerical setting; as such, the results produced by the classic algorithm in inexact arithmetic are but strong suggestions.

The experiments were performed on a computer system comprising two hexa-core processors and 48GB of main memory. Our code, including the QD library, was compiled with the GNU project C and C++ compiler version 4.4.3 with the flags `-O2 -fwhole-program` and `-frounding-math -mfpmath=sse -msse2`. The latter set of flags forces the compiler to generate machine code such

[12] We recognize that a more traditional implementation of the classic algorithm in a numerical setting computes the singular value decomposition and counts the number of sufficiently large singular values, e.g., as in (19); however, as there are no guarantees with respect to the correctness of the outcome, we have preferred, for the sake of this comparison, the fastest algorithm. We additionally performed experiments with the accurate `jacobiSvd` method provided by Eigen, but found that it was very slow compared to the approach based on the QR decomposition. The tolerance was arbitrarily chosen.

that all floating-point operations satisfy the correct rounding for single and double precision floating-point values, as specified in the IEEE 754 standard (33), by forcing the compiler to use the available SSE2 registers for performing floating-point arithmetic.[13] We additionally compiled the C version of W. Kahan's Paranoia program by **(author?)** (49), which attempts to verify the correct implementation of the IEEE 754 standard, with the same compiler and flags; running the program suggests that the target computer system indeed satisfies the IEEE 754 standard.

4.2 *A posteriori bound*

We consider here the numerical execution of Algorithm D. In Figure 1a, we visualize a typical progression of the three-term recurrence presented in Lemma 2.3 for a nondefective Segre variety. $\|A\mathbf{x}\|$ is the backward error, or residual, of the current approximation; a sufficiently large reduction of the initial residual is often used as stopping criterion for the CG method. The forward error related to applying the matrix-vector product, i.e., $\rho \approx 10^{-22}$, can be seen to be large with respect to the roundoff error, $\varepsilon \approx 10^{-32}$, but remains nearly constant. To be able to reject the hypothesis that G/D is singular, ω must be chosen larger than ρ. In practice, the attainable precision will be much less than ρ, as can be seen from χ in the figure. We note that χ increases most strongly whenever there is a peak for ϕ. These peaks are, in turn, caused by values of β strictly larger than 1 in subsequent iterations of the CG algorithm.

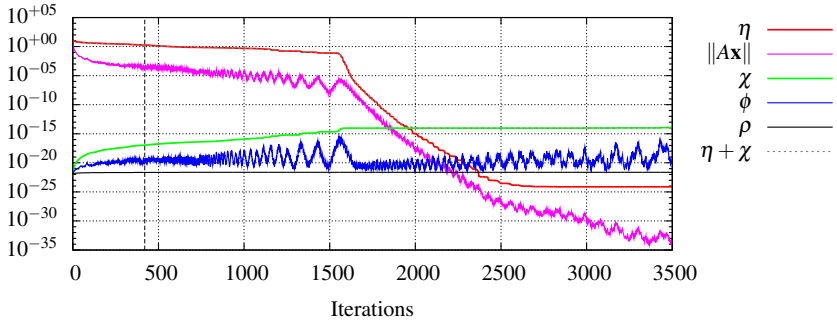
It can be observed in Figure 1a that the norm of the approximation, as well as the residual, hardly decreases during the first 1600 iterations, and then suddenly decays. This behavior occurs because the starting vector has standard normally distributed coefficients with respect to any basis for the matrix. Even in exact arithmetic, applying the CG method to G/D would only be able to reduce the residual to zero after $r(\Sigma - d + 1 - n_1)$ iterations. In addition, the loss of orthogonality in floating-point arithmetic causes a delay of convergence in the CG method (40). This unavoidable delay makes the choice of ω crucial to attain good performance; it will take at least $r(\Sigma - d + 1 - n_1)$ iterations before η will decrease significantly in magnitude—this minimum is indicated in Figure 1 by the vertical dashed line. On the other hand, when convergence sets in, we observed that it steadily progresses until the attainable accuracy is reached. While it takes 1422 iterations to reduce η to $1.00 \cdot 10^{-1}$, only 200 more iterations reduced it to $1.30 \cdot 10^{-5}$, and then 158 additional iterations reduce it to $1.10 \cdot 10^{-10}$ with another 224 iterations to get $1.00 \cdot 10^{-15}$, at which point the computation errors χ are larger, and the stopping criterion we proposed would have halted the algorithm. From this typical convergence behavior, it is clearly beneficial to choose ω as small as feasible, but not smaller; Algorithm S will only claim that G/D is nonsingular if it stops because $\chi + \eta < \omega$ for every tested starting vector.

In Figure 1b, the typical behavior of the three-term recurrence is displayed for a defective Segre variety in which neither D_1 or D_2' can be considered to be numerically singular. It shows that the residual can be decreased to approximately the roundoff error, while the norm of the solution remains of order one. The CG algorithm eventually terminates because the upper bound on the computation errors increases to the same order of magnitude as η, even though η remains practically unchanged.

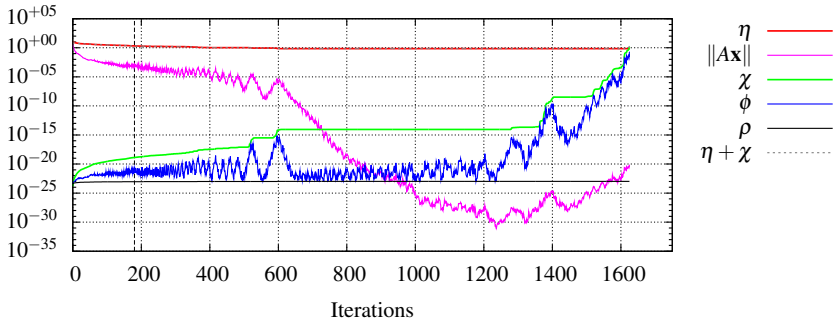
As we mentioned before, Lemma 2.3 is valid even when D_2' is (numerically) singular. As an illustration, consider the Segre variety $\mathbb{PC}^{20} \times \mathbb{PC}^5 \times \mathbb{PC}^2$, which is unbalanced defective. We find that D_2' can be considered numerically singular,[14] as its condition number is approximately $3.59 \cdot 10^{34}$ and $\|I - \mathrm{fl}((\underline{D_2'})^{-1})D_2'\| \leqslant 8.12 \cdot 10^{34}$. As a result, $\psi^{-1} \leqslant +\infty$ and Algorithm D immediately halts claiming there is insufficient evidence to conclude that $\mathscr{S}$ is nondefective.

[13] See http://gcc.gnu.org/wiki/x87note and http://gcc.gnu.org/wiki/FloatingPointMath.

[14] We know neither whether D_2' is mathematically singular, nor are we aware of any results in this direction.



(a) The nondefective Segre variety $\mathbb{PC}^{10} \times \mathbb{PC}^{9} \times \mathbb{PC}^{8}$.



(b) The defective Segre variety $\mathbb{PC}^{11} \times \mathbb{PC}^{11} \times \mathbb{PC}^{3}$.

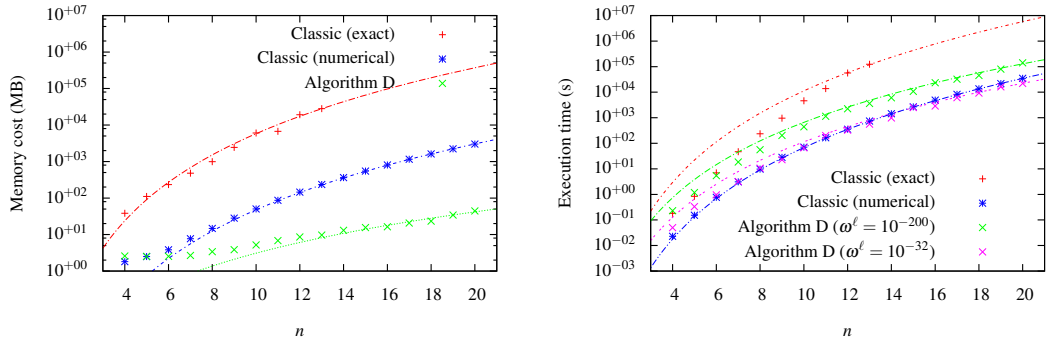
Figure 1: Typical evolution of the error estimates in the CG algorithm.

4.3 *Comparison with the classic algorithm*

In this subsection, Algorithm D is run with $\omega = 10^{-8}$. We tested both $\ell = 25$ and $\ell = 4$, so that the probability of incorrectly classifying a defective variety was, respectively, 10^{-200} and 10^{-32}. The memory cost and execution time of the classic and proposed algorithm in verifying the nondefectivity of the Segre varieties $\mathbb{PC}^n \times \mathbb{PC}^n \times \mathbb{PC}^n$, $4 \leqslant n \leqslant 20$ is investigated. The results are shown in Figure 2.

We start by verifying whether Algorithm D improves upon the memory cost of the classic algorithm by employing the structured matrix-vector product in Algorithm S. For the algorithms implemented in C++, we monitored the maximum memory consumption as reported by the operating system. The memory consumption of the classic algorithm in exact arithmetic was reported by Maple. The measured memory costs are displayed in Figure 2a; herein, the dashed lines represent the fitted theoretical complexities.[15] The classic algorithm in exact arithmetic could not be applied to Segre varieties with $n \geqslant 14$ as it required more main memory than the 48GB that was available. In inexact arithmetic, we find that it requires about 4.5, respectively 65, times more memory than Algorithm D for $n = 8$, respectively $n = 20$. Increasing the precision of the calculations would move the line corresponding to the classic algorithm in inexact arithmetic in Figure 2a upwards.

[15]These curves were obtained by the `fit` command in Gnuplot 4.2, where the only unknown we allowed was the constant masked by the big-O notation.



(a) Memory cost in MB

(b) Execution time in seconds

Figure 2: A comparison between the proposed and the classic algorithm for verifying the nondefectivity of the Segre variety $\mathbb{PC}^n \times \mathbb{PC}^n \times \mathbb{PC}^n$ in terms of execution time and memory consumption.

An indicative comparison of the execution times of the algorithms is presented in Figure 2b; the dashed lines indicate the fitted time complexities. We note that Algorithm D with $\omega^\ell = 10^{-32}$ is consistently competitive with the numerical implementation of the classic algorithm, which, we recall, does not provide a measure of likeliness that its answer is correct. Increasing the number of starting vectors ℓ causes the execution time of Algorithm D to increase proportionally.

Finally, we mention that all algorithms correctly reported that the considered Segre varieties are nondefective, which was already proved by **(author?)** (37).

4.4 *Results pertaining to Abo, Ottaviani, and Peterson's conjecture*

Because of the modest memory requirements of Algorithm D, we were able to concurrently verify nondefectiveness of 12 Segre varieties on the target computer system.[16] The algorithm was initially run with $\omega = 10^{-05}$ and $\ell = 11$, so that the probability of incorrectly classifying defective varieties is less than 10^{-55}. With these choices, *we verified that all balanced Segre varieties whose expected generic rank is less than or equal to* 55 *satisfy Conjecture 1.1*: no previously unknown defective Segre varieties were detected. Unfortunately, double–double arithmetic was not sufficiently accurate for handling all cases; in particular, the following cases could not be proved with double–double arithmetic:

$$53 \times 5 \times 3 \times 2 \times 2, \quad 54 \times 7 \times 3 \times 3, \quad 54 \times 18 \times 2 \times 2, \quad \text{and} \quad 55 \times 8 \times 2 \times 2 \times 2,$$

as well as

$$n \times n - 1 \times 2 \quad \text{with } n = 37, 40, 44, 47, 48, 50, 52, 53, 54, 55$$

Several perfect shapes also proved to be difficult; namely,

$$34 \times 21 \times 2 \times 2, \quad 51 \times 26 \times 3, \quad 53 \times 14 \times 5, \quad 53 \times 18 \times 2 \times 2, \quad 53 \times 27 \times 3, \quad \text{and} \quad 55 \times 28 \times 3,$$

and also

$$n \times n \times 2 \quad \text{with } 34 \leqslant n \leqslant 55 \quad \text{and} \quad n \neq 36, 38, 40, 53,$$

[16] Unfortunately, we ran out of processors long before we ran out of memory.

Note that all of these Segre varieties, except for $34 \times 21 \times 2 \times 2$, are nearly unbalanced defective. Running Algorithm D with quadruple–double arithmetic with $\omega = 10^{-11}$ and $\ell = 5$ allowed us to immediately verify the nondefectivity of aforementioned Segre varieties.

The nondefectivity results pertaining to Segre varieties with more than 4 factors are all original, excluding the known case of $(\mathbb{PC}^2)^d$, $d \geqslant 5$ (16). With the exception of the nondefectivity of $(\mathbb{PC}^n)^4$, $3 \leqslant n \leqslant 8$, and $n = 10$, recently proved by **(author?)** (21), the nondefectivity results for the 4-factor Segre variety are also new. In addition, we verified that both $(\mathbb{PC}^9)^4$ and $(\mathbb{PC}^{11})^4$ are nondefective (probability of incorrectly classifying a defective variety was less than 10^{-57}); two cases which could not be handled completely by the approach in (21).

5. Discussion

5.1 *Iterative methods for computing eigenvalues*

The reader may wonder why we did not consider an iterative eigenvalue algorithm for computing the smallest eigenvalue of an spsd matrix. For the sake of argument, let us consider the methods from (7, Section 4) and start with the simplest method: the power method. In our setting, one would have to use the shift-and-invert technique to converge to the smallest eigenvalue; by choosing a shift $\lambda > 0$, any eigenvalue zero corresponds to an eigenvalue λ^{-1} in $(A + \lambda I)^{-1}$. Suppose now that $A \in \mathbb{R}^{n \times n}$ is nonsingular with smallest eigenvalue ε, but that, unfortunately, we have chosen a shift $\lambda > \varepsilon = \delta\lambda$. This eigenvalue will be mapped to $\lambda^{-1}(1+\delta)^{-1}$; given that $\delta(1-\delta) < 1-(1+\delta)^{-1} < \delta$, we have that the relative distance to λ^{-1}, i.e., the remapped origin, is in the most optimistic scenario at most $\delta < 1$. Then, Theorem 4.1 from (34) can be used for obtaining an upper bound on the probability that after k steps we have converged to the largest eigenvalue $\lambda^{-1}(1+\delta)^{-1}$ with relative error less than δ, which is a lower bound on the least possible accuracy required for distinguishing between λ^{-1} and $\lambda^{-1}(1+\delta)^{-1}$: this probability would be bounded by $0.824\sqrt{n}(1-\delta)^{k-0.5}$. Note that if $\lambda > \varepsilon$, in the most favorable scenario the relative distance is 0.5, so one could substitute $\delta = 0.5$ in the previous bound. With the probability from (34)—whose result is far from trivial—one might then use it in a fashion similar to Algorithm S. The issue is that δ depends on λ and ε; the latter is obviously unknown. The problem could be resolved by obtaining (nontrivial) bounds on ε, but that would be as difficult as solving the original problem; that is, whether A is of full rank.

We deem it feasible to construct reliable methods based on the Lanczos method by considering the result in (35, Theorem 5). It would, however, still require n steps for an $n \times n$ matrix in exact arithmetic to conclude with probability one whether a matrix is nonsingular, which is precisely as costly as our Lemma 2.1; this comes as no surprise because the CG method is simply the Lanczos method for solving linear systems. The results by (34; 35) are not immediately applicable to inexact arithmetic, and we are not aware of any attempts of extending their work in this direction.

In regard to the Jacobi-Davidson method and restarted Lanczos method, we believe that, based on the above considerations, they would be considerably more difficult to analyze numerically to ensure that one converges to the smallest eigenvalue with a high *and computable* probability.

Because of the aforementioned issues, we are not convinced that eigenvalue algorithms would yield superior performance and their analysis is not expected to be simpler than the one presented in this paper; in particular, we are convinced that an analysis of the forward error as in Section 3.2 is unavoidable.

5.2 *Krylov subspace methods and the Gram matrix*

Another important concern is related to the use of the Gram matrix $G = Z^T Z$ in (3.2). Its condition number $\kappa(G)$ is the square of the condition number of Z. This may seem to adversely affect convergence, because the theoretical worst-case rate of convergence of the CG method applied to a matrix A at step k is, in the A-norm, proportional to $((\sqrt{\kappa(A)}-1)/(\sqrt{\kappa(A)}+1))^k$; see, e.g., (38, Section 5.6). However, this bound is only valid for the CG method, which cannot be applied to Z because it is not spsd: one must compare the rate of convergence of the CG method applied to G with an alternative Krylov method applied to Z. We observed empirically that Z is often strongly nonnormal, i.e., $\|ZZ^T - Z^T Z\|/\|Z\|$ is large, so that one would have to apply one of several Krylov subspace methods applicable to a general nonnormal matrix, such as, e.g., (restarted) GMRES, Bi-CGSTAB, or (transpose-free) QMR. Unfortunately, the theoretical convergence behavior of such methods applied to a strongly nonnormal matrix is more difficult to analyze; see, for instance, (38, Section 5.7.2–5.7.4) for some results about (unrestarted) GMRES. In exploratory experiments in floating-point arithmetic using Matlab 7.9.0, we observed for several (balanced) Segre varieties of small dimension that applying these iterative methods to Z was substantially less efficient than applying these methods, or any of the specialized methods for symmetric systems such as CG, MINRES and SYMMLQ, to G. Somewhat surprisingly, *the methods for general systems applied to Z mostly failed to converge within a reasonable number of steps, or even failed completely due to breakdown, while applying the same methods to G solved the system without problems*, albeit in more iterations than CG and MINRES. There are two additional computational reasons to prefer solving $G\mathbf{x} = 0$ over $Z\mathbf{x} = 0$. First, we were unable to derive a matrix-vector product with Z (and Z^T) that is as efficient as a matrix-vector product with G; the best method we were able to devise is asymptotically about d times more costly than a product with G. Second, preconditioners are, in practice, required to attain a fast rate of convergence. A preconditioner is a linear transformation applied to the system so that the transformed system is easier to solve using a Krylov subspace method. As Z is a structured matrix, it is highly advisable for computational reasons to devise a structured preconditioner with a cost that is roughly proportional to the cost of applying Z; however, no good preconditioners were obvious to us. For all of the above reasons, we decided that applying a general Krylov method to $Z\mathbf{x} = 0$ would be less effective than solving $G\mathbf{x} = 0$. For the sake of simplicity of our numerical analysis, we have chosen to employ the CG method, even though MINRES appeared to converge somewhat faster in our exploratory experiments.

In our exploratory experiments, we implemented and compared a few structured preconditioning methods for solving $G\mathbf{x} = 0$. The simplest preconditioner consisted of a block diagonal preconditioner that essentially exploits the Kronecker structure of (3.7) and (3.11) for explicitly computing the inverse of the blocks on the block diagonal and applying it to a vector. This preconditioner reduced the number of iterations, but not the total execution time, because of the added cost of applying it. We experimented with some form of structured Gaussian elimination based on Schur complements, where we maintained the structure of G, but this proved to be much too costly. In the end, we settled for a one-step version of the previous idea; while G/D was never introduced as such, it is in effect the resulting preconditioner. It has two interesting properties. First, there is a theoretical guarantee that this preconditioner reduces the condition number with respect to G because of the eigenvalue interlacing property of the Schur complement (45), which essentially reduces the magnitude of the largest eigenvalue. Second, it operates on the $r(\Sigma - n_1) \times r(\Sigma - n_1)$ matrix G/D, so that the theoretical number of iterations is now rn_1 less than for G, assuming we apply the CG method in exact arithmetic. Especially for Segre varieties whose dimension is close to being unbalanced, this reduction can be significant, and cause an iterative solution of G/D to converge in much fewer iterations than the same method applied to G.

6. Conclusions

We described a randomized algorithm for testing whether a matrix is nonsingular. The algorithm exploits the observation that the conjugate gradient method ignores the null space of the matrix in computing a solution to a linear system. We proved that, in a numerical setting, the components initially in the null space of the matrix can be bounded *a posteriori* by the norm of the obtained approximation plus the computation errors. Using this algorithm, we proposed a probabilistic algorithm, called Algorithm D, based on Terracini's lemma to check whether a given Segre variety is nondefective. While it cannot provide the certainty of a fully symbolical calculation, it offers flexibility in terms of execution time and has a significantly lower memory consumption, allowing it to tackle much larger problems than symbolical algorithms. Using our method, it was verified that all Segre varieties whose generic rank is smaller than 55 satisfy the conjecture proposed in **(author?)** (4); the probability that an individual defective variety was classified as nondefective is less than 10^{-55}.

The main feature of Algorithm D is that the probability of misclassifying a defective Segre variety can be reduced *without* increasing the precision of the calculations: the memory consumption is independent of the desired probability. This contrasts with a general perception in numerical algebraic geometry that the precision of the calculations must be increased to improve the "certainty" when using a numerical algorithm; we express hope that our algorithm may convince one that this is not always necessary.

The major issue with Algorithm D is the selection of the parameters ω and ℓ. Ideally, one chooses ω as small as possible; however, it appears to be difficult to assess what that constitutes. We observed that $\omega = 10^{-5}$ was often suitable, although some Segre varieties could be verified with $\omega = 10^{-8}$ or 10^{-12}. It appears that a higher rank entails higher computation errors, thus constraining the choice of ω.

There are several extensions of the proposed method that warrant further research. One promising direction consists of employing a block conjugate gradient method with block size ℓ. In this manner, the execution time of the algorithm may be reduced by a factor ℓ, because the Krylov space should be constructed only once for all ℓ vectors, whose support is noncompact, instead of the current approach in which a full Krylov space is constructed ℓ times. Our preliminary experiments suggest that using a block Krylov method may be a viable approach to reduce the execution time, while moderately increasing the memory requirements. A more involved numerical analysis would be required for establishing a bound on the probability of failure. Algorithm S can be applied to other structured matrices; however, a numerical analysis of the computation errors produced by the structured matrix-vector product would be required in each case. As an example, the techniques considered in this paper could be applied for verifying the nondefectivity of secant varieties of Segre–Veronese varieties.

Acknowledgements

We thank F. Tisseur for carefully handling our manuscript and her infallible judgment in selecting two helpful referees. The referees are thanked for their careful reading and many suggestions that considerably improved this manuscript, especially in Section 2, 3.2 and 5.

The first author is grateful to D. Abbeloos and B. Jeuris for the stimulating and encouraging discussions we shared; M. Ferranti is thanked for her critical remarks that substantially improved this paper.

Funding

The authors were supported by PFV/10/002 Optimization in Engineering (OPTEC), and the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office, Belgian Network

DYSCO (Dynamical Systems, Control, and Optimization). The first author's research was additionally supported by a Ph.D. fellowship of the Research Foundation—Flanders (FWO). The second author also acknowledges support by the Research Council KU Leuven project OT/11/055 (Spectral Properties of Perturbed Normal Matrices and their Applications), and by the Fund for Scientific Research–Flanders (Belgium) project G034212N (Reestablishing smoothness for matrix manifold optimization via resolution of singularities). The third author was, in addition, supported by the Research Council KU Leuven project OT/10/038 (Multi-parameter model order reduction and its applications).

REFERENCES

ABO, H., OTTAVIANI, G. & PETERSON, C. (2009) Induction for secant varieties of Segre varieties. *Trans. Amer. Math. Soc.*, **361**, 767–792.

ABO, H. (2010) On three conjectures about the secant defectivity of classically studied varieties. *Proceedings of the Algebraic Geometry Symposium*. Japan: Waseda University.

ABO, H., OTTAVIANI, G. & PETERSON, C. (2012) Non-defectivity of Grassmannians of planes. *J. Algebraic Geom.*, **21**, 1–20.

ABO, H. & BRAMBILLA, M. (2011) On the dimensions of secant varieties of Segre–Veronese varieties. *Ann. Mat. Pura Appl.*, 1–32.

ABO, H. & BRAMBILLA, M. (2012) New examples of defective secant varieties of Segre–Veronese varieties. *Collect. Math.*, **63**, 287–297.

ALEXANDER, J. & HIRSCHOWITZ, A. (1995) Polynomial interpolation in several variables. *J. Algebraic Geom.*, **4**, 201–222.

BAI, Z., DEMMEL, J., DONGARRA, J., RUHE, A. & VAN DER VORST, H. (eds) (2000) *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. Philadelphia, PA, USA: SIAM.

BATES, D. J., PETERSON, C. & SOMMESE, A. J. (2008) Application of a numerical version of Terracini's Lemma for secants and joins. *Algorithms in Algebraic Geometry* (A. Dickenstein, F.-O. Schreyer & A. J. Sommese eds). The IMA Volumes in Mathematics and its Applications, vol. 146. New York, NY, USA: Springer New York, pp. 1–14.

BRAMBILLA, M. & OTTAVIANI, G. (2008) On the Alexander–Hirschowitz theorem. *J. Pure Appl. Algebra*, **212**, 1229–1251.

CATALISANO, M., GERAMITA, A. & GIMIGLIANO, A. (2002) Ranks of tensors, secant varieties of Segre varieties and fat points. *Linear Algebra Appl.*, **355**, 263–285.

CATALISANO, M., GERAMITA, A. & GIMIGLIANO, A. (2005a) Higher secant varieties of Segre–Veronese varieties. *Projective varieties with unexpected properties: A volume in memory of Giuseppe Veronese: proceedings of the international conference "Varieties with Unexpected Properties," Siena, Italy, June 8–13, 2004* (C. Ciliberto, A. Geramita, B. Harbourne, R. Miró-Roig & K. Ranestad eds). Berlin, Germany: Walter de Gruyter GmbH & Co. KG, pp. 81–107.

CATALISANO, M., GERAMITA, A. & GIMIGLIANO, A. (2005b) Higher secant varieties of the Segre varieties $\mathbb{P}^1 \times \ldots \times \mathbb{P}^1$. *J. Pure Appl. Algebra*, **201**, 367–380.

CATALISANO, M., GERAMITA, A. & GIMIGLIANO, A. (2005c) Secant varieties of Grassmann varieties. *Proc. Amer. Math. Soc.*, **133**, 633–642.

CATALISANO, M., GERAMITA, A. & GIMIGLIANO, A. (2007) Segre–Veronese embeddings of $\mathbb{P}^1 \times \mathbb{P}^1 \times \mathbb{P}^1$ and their secant varieties. *Collect. Math.*, **1**, 1–24.

CATALISANO, M., GERAMITA, A. & GIMIGLIANO, A. (2008) On the ideals of secant varieties to certain rational varieties. *J. Algebra*, **319**, 1913–1931.

CATALISANO, M., GERAMITA, A. & GIMIGLIANO, A. (2011) Secant varieties of $\mathbb{P}^1 \times \cdots \times \mathbb{P}^1$ (n-times) are not defective for $n \geqslant 5$. *J. Algebraic Geom.*, **20**, 295–327.

CILIBERTO, C. & COOLS, F. (2008) On Grassmann secant extremal varieties. *Adv. Geom.*, **8**, 377–386.

COMON, P., TEN BERGE, J., DE LATHAUWER, L. & CASTAING, J. (2009) Generic and typical ranks of multi-way arrays. *Linear Algebra Appl.*, **430**, 2997–3007.

COMON, P. & MOURRAIN, B. (1996) Decomposition of quantics in sums of powers of linear forms. *Signal Processing*, **53**, 93–107.

EDELMAN, A. & RAO, N. (2005) Random matrix theory. *Acta Numer.*, **14**, 1–65.

GESMUNDO, F. (2013) An asymptotic bound for secant varieties of Segre varieties. *Ann. Univ. Ferrara*.

GOLUB, G. & STRAKOŠ, Z. (1994) Estimates in quadratic formulas. *Numer. Algorithms*, **8**, 241–268.

GREENBAUM, A. (1989) Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences. *Linear Algebra Appl.*, **113**, 7–63.

GREENBAUM, A. & STRAKOŠ, Z. (1992) Predicting the behavior of finite precision Lanczos and conjugate gradient computations. *SIAM J. Matrix Anal. Appl.*, **13**, 121–137.

GUENNEBAUD, G., JACOB, B. *et al.* (2010) *Eigen v3*. http://eigen.tuxfamily.org: http://eigen.tuxfamily.org.

GUTTMAN, L. (1946) Enlargement methods for computing the inverse matrix. *Ann. Math. Statist.*, **17**, 336–343.

HAYNSWORTH, E. (1968) Determination of the inertia of a partitioned Hermitian matrix. *Linear Algebra Appl.*, **1**, 73–81.

HESTENES, M. & STIEFEL, E. (1952) Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, **49**, 409–436.

HIDA, Y., LI, X. & BAILEY, D. (2001) Algorithms for quad-double precision floating point arithmetic. *Proceedings of the 15th IEEE Symposium on Computer Arithmetic*. Washington, DC, USA: IEEE Computer Society.

HIDA, Y., LI, X. & BAILEY, D. (2008) Library for double–double and quad–double arithmetic.

HIGHAM, N. (1996) *Accuracy and Stability of Numerical Algorithms*. Philadelphia, PA: SIAM.

HITCHCOCK, F. (1927) The expression of a tensor or a polyadic as a sum of products. *J. Math. Phys.*, **6**, 164–189.

IEEE (2008) *IEEE 754–2008 Standard for floating-point arithmetic*. *IEEE 754–2008 Standard for floating-point arithmetic*.

KUCZYŃSKI, J. & WOŹNIAKOWSKI, H. (1992) Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start. *SIAM J. Matrix Anal. Appl.*, **13**, 1094–1122.

KUCZYŃSKI, J. & WOŹNIAKOWSKI, H. (1994) Probabilistic bounds on the extremal eigenvalues and condition number by the Lanczos algorithm. *SIAM J. Matrix Anal. Appl.*, **15**, 672–691.

LANDSBERG, J. (2012) *Tensors: Geometry and Applications*. Graduate Studies in Mathematics, vol. 128. Providence, Rhode Island: American Mathematical Society.

LICKTEIG, T. (1985) Typical tensorial rank. *Linear Algebra Appl.*, **69**, 95–120.

LIESEN, J. & STRAKOŠ, Z. (2013) *Krylov Subspace Methods: Principles and Analysis*. Great Clarendon Street, Oxford, United Kingdom: Oxford University Press.

MCGILLIVRAY, B. (2006) A probabilistic algorithm for the secant defect of Grassmann varieties. *Linear Algebra Appl.*, **418**, 708–718.

MEURANT, G. & STRAKOŠ, Z. (2006) The Lanczos and conjugate gradient algorithms in finite precision arithmetic. *Acta Numer.*, **15**, 471–542.

NOTAY, Y. (1993) On the convergence of the conjugate gradients in presence of rounding errors. *Numer. Math.*, **65**, 301–317.

PAIGE, C. (1980) Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem. *Linear Algebra Appl.*, **34**, 235–258.

RUMP, S. (2011) Verified bounds for singular values, in particular for the spectral norm of a matrix and its inverse. *BIT*, **51**, 367–384.

SAAD, Y. (2003) *Iterative Methods for Sparse Linear Systems*, 2nd edn. Philadelphia, PA, USA: SIAM.

SMITH, R. L. (1992) Some interlacing properties of the Schur complement of a hermitian matrix. *Linear Algebra Appl.*, **177**, 137–144.

STRAKOŠ, Z. (1991) On the real convergence rate of the conjugate gradient method. *Linear Algebra Appl.*, **154–156**, 535–549.

STRAKOŠ, Z. & TICHÝ, P. (2002) On error estimation in the conjugate gradient method and why it works in finite precision computations. *Electron. Trans. Numer. Anal.*, **13**, 56–80.

STRASSEN, V. (1983) Rank and optimal computation of generic tensors. *Linear Algebra Appl.*, **52–53**, 645–685.

SUMNER, T. & GAY, D. (1992) A C version of Kahan's floating point test "paranoia".

TERRACINI, A. (1911) Sulla V_k per cui la varietà degli S_h $h+1$-secanti ha dimensione minore dell'ordinario. *Rend. Circ. Mat. Palermo*, **31**, 392–396.

VAN LOAN, C. (2000) The ubiquitous Kronecker product. *J. Comput. Appl. Math.*, **123**, 85–100.

VANNIEUWENHOVEN, N., VANDEBRIL, R. & MEERBERGEN, K. (2012) A probabilistic numerical algorithm based on Terracini's lemma for proving nondefectivity of Segre varieties. *Tech. Rep.* TW622. Leuven, Belgium: KU Leuven.

WILKINSON, J. H. (1965) *The Algebraic Eigenvalue Problem*. Amen House, London E.C. 4, United Kingdom: Oxford University Press.

YANG, M. (2013) On partial and generic uniqueness of block term tensor decompositions. *Ann. Univ. Ferrara*, 1–29.

ZAK, F. L. (1993) *Tangents and Secants of Algebraic Varieties*. Translations of mathematical monographs, vol. 127. Providence, Rhode Island: AMS.

A. Proof of the lemmas

Proof of Lemma 3.1. Consider the tautology $\underline{Z}\,\underline{Y} - ZY = \underline{X} - X + E_b(\underline{X},\underline{Y},\underline{Z})$ that can be written as

$$(\underline{Z}-Z)Y = E_X + E_b(\underline{X},\underline{Y},\underline{Z}) - \underline{Z}E_Y, \quad \text{or, equivalently,} \quad E_Z = (E_X + E_b(\underline{X},\underline{Y},\underline{Z}) - \underline{Z}E_Y)Y^{-1}.$$

Proceed by taking the max-norm and bounding $\|Y^{-1}\|_1$ using Lemma 2.1 of **(author?)** (43):

$$\frac{1}{\sqrt{n}}\|Y^{-1}\|_1 \leqslant \|Y^{-1}\|_2 \leqslant \frac{\left\|\mathrm{fl}\left(\underline{Y}^{-1}\right)\right\|_2}{1-\alpha}$$

with $\alpha > 1$ and

$$\alpha = \left\|I - \mathrm{fl}\left(\underline{Y}^{-1}\right)Y\right\|_2 \leqslant \left\|I - \mathrm{fl}\left(\underline{Y}^{-1}\right)(\underline{Y}-E_Y)\right\|_F \leqslant \left\|I - \mathrm{fl}\left(\underline{Y}^{-1}\right)\underline{Y}\right\|_F + \|\,\mathrm{fl}\left(\underline{Y}^{-1}\right)\|_F \|E_Y\|_F.$$

The condition $1-\alpha > 0$ may be discarded by considering ψ as in the lemma, because $+\infty$ is always a valid upper bound on $\|\cdot\|_1$.

If Y is singular, we have by Lemma 2.1 of **(author?)** (43) that $1-\alpha < 0$ so that the thesis follows. □

Proof of Lemma 3.2. The inner product matrices satisfy

$$\underline{J}^{(k)} = (A^{(k)})^T A^{(k)} + E_J^{(k)} \quad \text{with} \quad |E_J^{(k)}| \leqslant \varepsilon_{n_k},$$

observing that every entry of $|A^{(k)}|^T|A^{(k)}|$ is the inner product of two unit norm vectors so that the Cauchy–Schwarz inequality bounds it by the norm of these vectors. From the previous equation we find

$$\underline{H}^{(i,j)} := \mathrm{fl}\left(\bigodot_{\substack{k=1\\k\neq i,j}}^{d} \underline{J}^{(k)}\right) = \bigodot_{\substack{k=1\\k\neq i,j}}^{d}\left(J^{(k)} + E_J^{(k)}\right) + E_{\odot J'}^{(i,j)}, \text{ with } |E_{\odot J'}^{(i,j)}| \leqslant \varepsilon_{d-3} \bigodot_{\substack{k=1\\k\neq i,j}}^{d} |\underline{J}^{(k)}| \leqslant \varepsilon_{d-3}, \quad \text{(A.1)}$$

where we observe that even in floating-point arithmetic the product of values strictly smaller than 1 remains smaller than 1. Recalling that $n_1 = \max_k\{n_k\}$, we can bound

$$\left| \bigodot_{\substack{k=1 \\ k\neq i,j}}^{d} \left(J^{(k)} + E_J^{(k)}\right) - \bigodot_{\substack{k=1 \\ k\neq i,j}}^{d} J^{(k)} \right| \leqslant \sum_{i=1}^{d-2} \binom{d-2}{i} (\varepsilon_{n_1})^i = (1+\varepsilon_{n_1})^{d-2} - 1 \leqslant \varepsilon_{(d-2)n_1}, \tag{A.2}$$

where the first inequality is obtained from

$$\left| \prod_{i=1}^{k} (a_i + b_i) - \prod_{i=1}^{k} a_i \right| \leqslant \sum_{i=1}^{k} \binom{k}{i} a^{k-i} b^i,$$

where $|a_i| \leqslant a$ and $|b_i| \leqslant b$ for all i, and using that $|j_{rs}^{(k)}| \leqslant 1$ and $|(E_J^{(k)})_{rs}| \leqslant \varepsilon_{n_1}$. Using (3.20), we find that (A.1) plus (A.2) is bounded by $\varepsilon_{(d-2)(n_1+1)}$, and consequently,

$$\underline{H}^{(i,j)} = H^{(i,j)} + E_{\odot J}^{(i,j)}, \quad \text{where} \quad |E_{\odot J}^{(i,j)}| \leqslant \varepsilon_{(d-2)(n_1+1)}. \tag{A.3}$$

Let $K := (A^{(j)})^T X$, then we obtain

$$\underline{K} := \mathrm{fl}((A^{(j)})^T \underline{X}) = (A^{(j)})^T X + (A^{(j)})^T E_X + E_{K'} := K + E_K, \tag{A.4}$$

where, by the product rule (3.19),

$$|E_{K'}| \leqslant \varepsilon_{n_j} |A^{(j)}|^T |\underline{X}| \leqslant \varepsilon_{n_j} \|(A^{(j)})^T\|_\infty \|\underline{X}\|_{\max} = \varepsilon_{n_j} \|A^{(j)}\|_1 \|\underline{X}\|_{\max},$$

and, thus,

$$|E_K| \leqslant |E_{K'}| + |A^{(j)}|^T |E_X| \leqslant \varepsilon_{n_j} \|A^{(j)}\|_1 \|\underline{X}\|_{\max} + \|A^{(j)}\|_1 \|E_X\|_{\max} \leqslant \varepsilon_{n_j} \alpha_j \|X\|_{\max} + \alpha_j (1+\varepsilon_{n_j}) \varepsilon_X.$$

Then, combining (A.3) and (A.4),

$$\begin{aligned} \underline{H}_{\underline{X}}^{(i,j)} &= \mathrm{fl}\left((K + E_K) \odot (E_{\odot J}^{(i,j)} + H^{(i,j)}) \right) \\ &= H_X^{(i,j)} + E_K \odot H^{(i,j)} + E_{\odot J}^{(i,j)} \odot K + E_K \odot E_{\odot J}^{(i,j)} + E_{K \odot J}^{(i,j)} := H_X^{(i,j)} + E_{H_X}^{(i,j)}. \end{aligned} \tag{A.5}$$

We can then bound

$$|E_{H_X}^{(i,j)}| \leqslant \left(|E_K| + |K| \odot |E_{\odot J}^{(i,j)}| + |E_K| \odot |E_{\odot J}^{(i,j)}| \right) + |E_{K \odot J}^{(i,j)}|.$$

The last term in the above expression is bounded by ε times the bracketed expression plus $\varepsilon |H_X^{(i,j)}|$, which can be verified through straightforward computations. Furthermore,

$$|H_X^{(i,j)}| = |(A^{(j)})^T X| \odot |H^{(i,j)}| \leqslant |A^{(j)}|^T |X| \leqslant \|A^{(j)}\|_1 \|X\|_{\max}. \tag{A.6}$$

We thus obtain

$$\begin{aligned} |E_{H_X}^{(i,j)}| &\leqslant (1+\varepsilon) \left(|E_K| + |K| \odot |E_{\odot J}^{(i,j)}| + |E_K| \odot |E_{\odot J}^{(i,j)}| \right) + \varepsilon |H_X^{(i,j)}| \\ &\leqslant (1+\varepsilon) \left((1+\varepsilon_{(d-2)(n_1+1)}) |E_K| + \varepsilon_{(d-2)(n_1+1)} \alpha_j \|X\|_{\max} \right) + \varepsilon \alpha_j \|X\|_{\max} \end{aligned}$$

Algorithm 3: Computing $(D/D_1)^{-1}$ and L_2'

input : $\{A^{(k)}\}_k$, $\{J^{(k)}\}_k$.
output: L_2' and D_2' as in (3.16).

1: $L_{1,2}^\star \leftarrow T_R^T T^1$
2: $L_2^\star \leftarrow \begin{bmatrix} T_R^T T^2 & T_R^T T^3 & \cdots & T_R^T T^d \end{bmatrix}^T$
3: $D_2 \leftarrow T_R^T T_R$
4: $H^{(1)} \leftarrow \bigodot_{k=2}^d J^{(k)}$
5: **for** $i = 1,2,\ldots,\Sigma - d + 1$ **do**
6: $\quad \dot{L}_{\{i\}} \leftarrow L_{\{i\}}^\star (H^{(1)})^{-1}$
7: **end**
8: **for** $j = 2,3,\ldots,d$ **do**
9: $\quad H \leftarrow \bigodot_{k=2,k\neq j} J^{(k)}$
10: $\quad$ **for** $i = 1,2,\ldots,\Pi - r(\Sigma - d + 1)$ **do**
11: $\quad\quad H_{\dot{L}_{\{i\}}}^{(j,1)} \leftarrow ((A^{(1)})^T \dot{L}_{\{i\}}) \odot H$
12: $\quad\quad (\ddot{\mathbf{I}}_i)_j \leftarrow \mathrm{vec}(A^{(j)} H_{\dot{L}_{\{i\}}}^{(j,1)})$
13: $\quad$ **end**
14: **end**
15: $L_2' \leftarrow (L_2^\star - \ddot{L}) P_R C_R$
16: $D_2' \leftarrow C_R^T P_R^T ((D_2 - L_{1,2}^\star \dot{L}) P_R C_R)$

$$\begin{aligned}
&\leqslant (1+\varepsilon)\left(\varepsilon_{(d-2)(n_1+1)+n_j}\alpha_j\|X\|_{\max} + (1+\varepsilon_{(d-2)(n_1+1)})(1+\varepsilon_{n_j})\alpha_j\varepsilon_X\right) + \varepsilon\alpha_j\|X\|_{\max} \\
&\leqslant \varepsilon_{(d-1)(n_1+1)}\alpha_j\|X\|_{\max} + (1+\varepsilon_{(d-1)(n_1+1)})\alpha_j\varepsilon_X,
\end{aligned}$$

noting that we bounded ε_{n_j} by ε_{n_1} in the penultimate step, concluding the proof. □

Proof of Lemma 3.3. The forward error on the computations in lines 1 through 3 of Algorithm 3 can be handled simultaneously; we have $\mathrm{fl}((T_R^i)^T T_k^j) = (T_R^i)^T T_k^j + E_{T^{(i,j)}}$ where

$$|E_{T^{(i,i)}}| \leqslant \varepsilon_{(d-2)(n_1+1)} \quad \text{and} \quad |E_{T^{(i,j)}}| \leqslant \varepsilon_{(d-1)(n_1+1)};$$

the first bound follows readily from (A.3), and the second bound exploits the observation that the round-off error for computing $\mathbf{a}_k^i(\mathbf{a}_R^j)^T$ is bounded by ε, because the vectors $\mathbf{a}$ are normalized. Concluding:

$$\begin{aligned}
&\underline{L}_{1,2}^\star = L_{1,2}^\star + E_{L_{1,2}^\star}, && \underline{L}_2^\star = L_2^\star + E_{L_2^\star}, \text{ and} && \underline{D}_2 = D_2 + E_{D_2}, \quad \text{with} \\
&|E_{L_{1,2}^\star}| \leqslant \varepsilon_{(d-1)(n_1+1)}, && |E_{L_2^\star}| \leqslant \varepsilon_{(d-1)(n_1+1)}, \text{ and} && |E_{D_2}| \leqslant \varepsilon_{(d-1)(n_1+1)}.
\end{aligned}$$

A reasoning similar to the derivation of (A.3) in the proof of Lemma 3.2 yields the error in line 4:

$$\underline{H}^{(1)} = H^{(1)} + E_H^{(1)} \quad \text{with} \quad |E_H^{(1)}| \leqslant \varepsilon_{(d-1)(n_1+1)}.$$

The forward error $E_{\dot{L}_{\{i\}}} = \dot{L}_{\{i\}} - \underline{\dot{L}}_{\{i\}}$ in line 6 satisfies

$$\begin{aligned}
|E_{\dot{L}_{\{i\}}}| &\leqslant \tau_1\left(\|E_b(\underline{L}_{\{i\}}^\star, \underline{H}^{(1)}, \underline{\dot{L}}_{\{i\}})\|_{\max} + \varepsilon_{(d-1)(n_1+1)} + \|\underline{\dot{L}}_{\{i\}} E_H^{(1)}\|_{\max}\right), \\
&\leqslant \tau_1\varepsilon_{(d-1)(n_1+1)}\left(\varepsilon_{(d-1)(n_1+1)}^{-1}\|E_b(\underline{L}_{\{i\}}^\star, \underline{H}^{(1)}, \underline{\dot{L}}_{\{i\}})\|_{\max} + 1 + \|\underline{\dot{L}}_{\{i\}}\|_\infty\right) \leqslant \tau_1\varepsilon_{\mu(d-1)(n_1+1)},
\end{aligned}$$

in which the first inequality follows immediately from an application of Lemma 3.1.

Lines 9 and 11 are covered by Lemma 3.2; applying it yields:

$$|E^{(j,1)}_{H_{\dot{L}_{\{i\}}}}| \leqslant \varepsilon_{(d-1)(n_1+1)}\alpha_1\|\dot{L}_{\{i\}}\|_{\max} + (1+\varepsilon_{(d-1)(n_1+1)})\alpha_1\tau_1\varepsilon_{\mu(d-1)(n_1+1)} \leqslant \alpha_1\tau_1\varepsilon_{(\mu+1)(d-1)(n_1+1)},$$

where we noted that $\|\dot{L}_{\{i\}}\|_{\max} \leqslant \|L^{\star}_{\{i\}}\|_{\max}\|(H^{(1)})^{-1}\|_1 \leqslant \tau_1$.

Line 12 computes $\underline{\ddot{L}}^{(j)}_{\{i\}} := \mathrm{fl}(A^{(j)}\underline{H}^{(j,1)}_{\dot{L}_{\{i\}}}) = A^{(j)}H^{(j,1)}_{\dot{L}_{\{i\}}} + A^{(j)}E^{(j,1)}_{H_{\dot{L}_{\{i\}}}} + E'^{(j)}_{\dot{L}_{\{i\}}} := \ddot{L}_{\{i\}} + E^{(j)}_{\ddot{L}_{\{i\}}}$, in which

$$\begin{aligned}|E'^{(j)}_{\dot{L}_{\{i\}}}| &\leqslant \varepsilon_r\|A^{(j)}\|_{\infty}\left(\left\|H^{(j,1)}_{\dot{L}_{\{i\}}}\right\|_{\max} + \left\|E^{(j,1)}_{H_{\dot{L}_{\{i\}}}}\right\|_{\max}\right)\\ &\leqslant \varepsilon_r\|A^{(j)}\|_{\infty}\left(\|(A^{(1)})^T\dot{L}_{\{i\}}\|_{\max} + \alpha_1\tau_1\varepsilon_{(\mu+1)(d-1)(n_1+1)}\right),\end{aligned}$$

so that the forward error is bounded by

$$|E^{(j)}_{\ddot{L}_{\{i\}}}| \leqslant \|A^{(j)}\|_{\infty}\alpha_1\tau_1(\varepsilon_{(\mu+1)(d-1)(n_1+1)} + \varepsilon_r(1+\varepsilon_{(\mu+1)(d-1)(n_1+1)})) \leqslant \|A^{(j)}\|_{\infty}\alpha_1\tau_1\varepsilon_{r+(\mu+1)(d-1)(n_1+1)}.$$

Taking the maximum over $2 \leqslant j \leqslant d$ and $1 \leqslant i \leqslant \Sigma - d + 1$ allows us to write $\underline{\ddot{L}} = \ddot{L} + E_{\ddot{L}}$, where

$$|E_{\ddot{L}}| \leqslant \tau_1\alpha_1\varepsilon_{r+(\mu+1)(d-1)(n_1+1)}\max_{2\leqslant j\leqslant d}\|A^{(j)}\|_{\infty}.$$

Note that $|\ddot{L}| \leqslant \|A^{(j)}\|_{\infty}\|(A^{(1)})^T\dot{L}_{\{i\}}\|_{\max} \leqslant \alpha^{\infty}\alpha_1\tau_1$.

Line 15 of Algorithm 3 satisfies: $\underline{L}''_2 := \mathrm{fl}\left(\underline{L}^{\star}_2 - \underline{\ddot{L}}\right) = L^{\star}_2 - \ddot{L} + E_{L^{\star}_2} - E_{\ddot{L}} + E'_{L''_2} := L'_2 + E_{L''_2}$, where a brief calculation reveals

$$|E'_{L''_2}| \leqslant \varepsilon\left(|\underline{L}^{\star}_2| + |\underline{\ddot{L}}|\right) \leqslant \varepsilon(|L^{\star}_2| + |\ddot{L}| + |E_{L^{\star}_2}| + |E_{\ddot{L}}|).$$

Because $|L^{\star}_2| \leqslant 1$ and $|\ddot{L}| \leqslant \alpha^{\infty}\alpha_1\tau_1$, we obtain

$$\begin{aligned}|E_{L''_2}| &\leqslant \varepsilon(1+\alpha^{\infty}\alpha_1\tau_1) + (1+\varepsilon)(\varepsilon_{(d-1)(n_1+1)} + \alpha^{\infty}\alpha_1\tau_1\varepsilon_{r+(\mu+1)(d-1)(n_1+1)})\\ &\leqslant \varepsilon_{r+1+(\mu+1)(d-1)(n_1+1)}\alpha^{\infty}\alpha_1\tau_1 + \varepsilon_{1+(d-1)(n_1+1)}\\ &\leqslant \varepsilon_{r+1+(\mu+1)(d-1)(n_1+1)}(1+\alpha^{\infty}\alpha_1\tau_1),\end{aligned}$$

and, furthermore, $|L''_2| \leqslant |L^{\star}_2| + |\ddot{L}| \leqslant 1 + \alpha^{\infty}\alpha_1\tau_1$.

We note that applying P_R introduces no roundoff error, as we simply select some columns of $\underline{L}''_2$. The forward error of the computation of L'_2 is obtained as follows:

$$\underline{L}'_2 := \mathrm{fl}\left(\underline{L}''_2P_RC_R\right) = \underline{L}''_2P_RC_R + E'_{L'_2} = L'_2 + E_{L''_2}P_RC_R + E'_{L'_2} = L'_2 + E_{L'_2},$$

which we can bound by

$$\begin{aligned}|E_{L'_2}| &\leqslant (1+\alpha^{\infty}\alpha_1\tau_1)\|P_RC_R\|_1\varepsilon_{r+1+(\mu+1)(d-1)(n_1+1)} + \varepsilon_{c'}\left(\|L''_2\|_{\max} + \|E_{L''_2}\|_{\max}\right)\|P_RC_R\|_1\\ &\leqslant (1+\alpha^{\infty}\alpha_1\tau_1)\|C_R\|_1\left((1+\varepsilon_{c'})\varepsilon_{r+1+(\mu+1)(d-1)(n_1+1)} + \varepsilon_{c'}\right)\\ &\leqslant (1+\alpha^{\infty}\alpha_1\tau_1)\|C_R\|_1\varepsilon_{c'+r+1+(\mu+1)(d-1)(n_1+1)},\end{aligned}$$

where we observed $\|P_RC_R\|_1 \leqslant \|P_R\|_{\max}\|C_R\|_1$ and P_R is a binary matrix. This provides a bound on the forward error of the computation of L'_2 by Algorithm 3.

The error for the computation of D_2' in line 16 can be derived. Consider $\underline{\dot{D}}_2 := \mathrm{fl}(\underline{L}_{1,2}^\star \underline{\dot{L}}) = L_{1,2}^\star \dot{L} + E_{\dot{D}_2}$, where the error term may be bounded as

$$\begin{aligned}
|E_{\dot{D}_2}| &\leqslant |\underline{L}_{1,2}^\star E_{\dot{L}}| + |E_{L_{1,2}^\star} \dot{L}| + \varepsilon_{n_1 r} |\underline{L}_{1,2}^\star||\underline{\dot{L}}| \\
&\leqslant \|\underline{L}_{1,2}^\star\|_\infty \left(\tau_1 \varepsilon_{\mu(d-1)(n_1+1)} + \varepsilon_{n_1 r}(\|\dot{L}\|_{\max} + \|E_{\dot{L}}\|_{\max}\right) + \|E_{L_{1,2}^\star}\|_\infty \|\dot{L}\|_{\max} \\
&\leqslant \|\underline{L}_{1,2}^\star\|_\infty \tau_1 \left(\varepsilon_{\mu(d-1)(n_1+1)} + \varepsilon_{n_1 r}(1 + \varepsilon_{\mu(d-1)(n_1+1)})\right) + n_1 r \varepsilon_{(d-1)(n_1+1)} \tau_1 \\
&\leqslant n_1 r \tau_1 \varepsilon_{n_1 r + (\mu+1)(d-1)(n_1+1)},
\end{aligned}$$

where in the last step we observed $\|\underline{L}_{1,2}^\star\|_\infty \leqslant n_1 r \|\underline{L}_{1,2}^\star\|_{\max}$ and that even in floating-point arithmetic the product of numbers smaller than one remains bounded by one. Then, $\underline{\ddot{D}}_2 := \mathrm{fl}\left(\underline{D}_2 - \underline{\dot{D}}_2\right) = D_2 - \dot{D}_2 + E_{\ddot{D}_2} := \ddot{D}_2 + E_{\ddot{D}_2}$, where the forward error is bounded by

$$\begin{aligned}
|E_{\ddot{D}_2}| &\leqslant (1+\varepsilon)(|E_{D_2}| + |E_{\dot{D}_2}|) + \varepsilon(|D_2| + |\dot{D}_2|) \\
&\leqslant (1+\varepsilon)(\varepsilon_{(d-1)(n_1+1)} + |E_{\dot{D}_2}|) + \varepsilon(1 + n_1 r \tau_1) \\
&\leqslant \varepsilon_{1+(d-1)(n_1+1)} + n_1 r \tau_1 \varepsilon_{n_1 r + 1 + (\mu+1)(d-1)(n_1+1)},
\end{aligned}$$

where in the first inequality we observed that $|\dot{D}_2| \leqslant \|L_{1,2}^\star\|_\infty \|\dot{L}\|_{\max} \leqslant n_1 r \|L_{1,2}^\star\|_{\max} \|\dot{L}\|_{\max} \leqslant n_1 r \tau_1$. Consequently, $|\ddot{D}| \leqslant 1 + n_1 r \tau_1$. Proceeding with the matrix multiplication, we write $\underline{\dddot{D}}_2 := \mathrm{fl}\left(\underline{\ddot{D}}_2 P_R C_R\right) = \ddot{D}_2 P_R C_R + E_{\dddot{D}_2}$, where the forward error obeys

$$\begin{aligned}
|E_{\dddot{D}_2}| &\leqslant |E_{\ddot{D}_2}||P_R C_R| + \varepsilon_{c'} |\underline{\ddot{D}}_2||P_R C_R| \\
&\leqslant \left((1+\varepsilon_{c'})\|E_{\ddot{D}_2}\|_{\max} + \varepsilon_{c'}(\|D_2\|_{\max} + \|\dot{D}_2\|_{\max})\right)\|C_R\|_1 \\
&\leqslant \left((1+\varepsilon_{c'})\|E_{\ddot{D}_2}\|_{\max} + \varepsilon_{c'}(1 + n_1 r \tau_1)\right)\|C_R\|_1 \\
&\leqslant \left(\varepsilon_{c'+1+(d-1)(n_1+1)} + n_1 r \tau_1 \varepsilon_{c'+n_1 r+1+(\mu+1)(d-1)(n_1+1)}\right)\|C_R\|_1.
\end{aligned}$$

Finally, the last matrix product may be written as $\underline{D}_2' = \mathrm{fl}\left(C_R^T P_R^T \underline{\dddot{D}}_2\right) = C_R^T P_R^T \dddot{D}_2 + E_{D_2'}$, in which the error term is similarly bounded:

$$\begin{aligned}
|E_{D_2'}| &\leqslant |P_R C_R|^T |E_{\dddot{D}_2}| + \varepsilon_{c'} |P_R C_R|^T |\underline{\dddot{D}}_2| \\
&\leqslant \left((1+\varepsilon_{c'})\|E_{\dddot{D}_2}\|_{\max} + \varepsilon_{c'}\|\ddot{D}_2 P_R C_R\|_{\max}\right)\|C_R\|_1 \\
&\leqslant \left((1+\varepsilon_{c'})\|E_{\dddot{D}_2}\|_{\max} + \varepsilon_{c'}(1 + n_1 r \tau_1)\|C_R\|_1\right)\|C_R\|_1 \\
&\leqslant \left(\varepsilon_{2c'+1+(d-1)(n_1+1)} + n_1 r \tau_1 \varepsilon_{2c'+n_1 r+1+(\mu+1)(d-1)(n_1+1)}\right)\|C_R\|_1^2.
\end{aligned}$$

This concludes the derivation. □