Algorithm xxx: Improved invariant polytope algorithm and applications

THOMAS MEJSTRIK, University of Vienna, Austria

In several papers of 2013 – 2016, Guglielmi and Protasov made a breakthrough in the problem of the joint spectral radius computation, developing the invariant polytope algorithm which for most matrix families finds the exact value of the joint spectral radius. This algorithm found many applications in problems of functional analysis, approximation theory, combinatorics, etc.. In this paper we propose a modification of the invariant polytope algorithm making it roughly 3 times faster (single threaded), suitable for higher dimensions and parallelise it. The modified version works for most matrix families of dimensions up to 25, for non-negative matrices up to 3000. Besides we introduce a new, fast algorithm, called modified Gripenberg algorithm, for computing good lower bounds for the joint spectral radius. The corresponding examples and statistics of numerical results are provided. Several applications of our algorithms are presented. In particular, we find the exact values of the regularity exponents of Daubechies wavelets up to order 42 and the capacities of codes that avoid certain difference patterns.

CCS Concepts: • **Mathematics of computing → Computations on matrices**; *Mathematical software performance*; • **Computing methodologies** → *Parallel computing methodologies.*

Additional Key Words and Phrases: joint spectral radius, invariant polytope algorithm, parallelization, Daubechies wavelets, capacity of codes, norm estimation

ACM Reference Format:
Thomas Mejstrik. 2020. Algorithm xxx: Improved invariant polytope algorithm and applications. *ACM Trans. Math. Softw.* 1, 1 (June 2020), 26 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION AND NOTATION

The *joint spectral radius* (JSR) of a set of matrices is a quantity which describes the maximal asymptotic growth rate of the norms of products of matrices from that set (with repetitions permitted). Precisely, given a finite set $\mathcal{A} = \{A_j : j = 1, \dots, J\} \subseteq \mathbb{R}^{s\times s}$, $s \in \mathbb{N}$, then

$$\mathrm{JSR}(\mathcal{A}) := \lim_{n\to\infty} \max_{A_j\in\mathcal{A}} \left\|A_{j_n}\cdots A_{j_2}A_{j_1}\right\|^{1/n}. \tag{1}$$

In [3] it is proved that (for finite $\mathcal{A}$)

$$\mathrm{JSR}(\mathcal{A}) = \limsup_{n\to\infty,\ A_j\in\mathcal{A}} \rho(A_{j_n}\cdots A_{j_2}A_{j_1})^{1/n}, \tag{2}$$

where ρ is the classical *spectral radius* of a matrix. With $\#\mathcal{A}$ we denote the *number of elements* of the set $\mathcal{A}$. If $\#\mathcal{A} = 1$, then the JSR reduces to the spectral radius of a matrix.

The JSR has been defined in [31] and since appeared in many (seemingly unrelated) mathematical applications, e.g. for computing the regularity of wavelets and of subdivision schemes [15], the

Author's address: Thomas Mejstrik, Faculty of Mathematics, University of Vienna, Universitätsring 1, 1010 Vienna, Austria, tommsch@gmx.at.

0098-3500/2020/6-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

capacity of codes [28], the stability of linear switched systems [23] or in connection with the Euler partition function [30].

The computation of the JSR is a notoriously hard problem. Even for non-negative matrices with rational coefficients this problem is NP-hard [10]. Moreover, the question whether $\mathrm{JSR}(\mathcal{A}) \leq 1$ for a given set $\mathcal{A}$ is algorithmically undecidable [11]. Most algorithms which try to compute or to approximate the JSR make use of the inequality [15]

$$\max_{A_j \in \mathcal{A}} \rho\left(A_{j_k} \cdots A_{j_1}\right)^{1/k} \leq \mathrm{JSR}(\mathcal{A}) \leq \max_{A_j \in \mathcal{A}} \left\|A_{j_k} \cdots A_{j_1}\right\|^{1/k}, \tag{3}$$

which holds for any $k \in \mathbb{N}$. For a product $A_{j_k} \cdots A_{j_1}$ we say the number $\rho\left(A_{j_k} \cdots A_{j_1}\right)^{1/k}$ is its *normalized spectral radius*. Equation (3) tells us that the normalized spectral radius of every product is a valid lower bound for the JSR, on the contrary, one has to compute the norms of all products of a fixed length $k \in \mathbb{N}$ to obtain a valid upper bound.

If there exists a product $\Pi = A_{j_n} \cdots A_{j_1}, A_j \in \mathcal{A}$, such that $\rho(\Pi)^{1/n} = \mathrm{JSR}(\mathcal{A})$, we call the product a *spectral maximizing product (s.m.p.)*. It has been shown that there exist sets of matrices such that the normalized spectral radius of every finite product is strictly less than the JSR [24]. In other words, not all sets of matrices posses an s.m.p.. It is an open question whether pairs of binary matrices always posses an s.m.p. [6].

An s.m.p. is called *dominant* if there exists $\gamma > 0$ such that $\gamma < \mathrm{JSR}(\mathcal{A})$ and $\rho(A_{j_l} \cdots A_{j_1})^{1/l} < \gamma$ whenever $A_{j_l} \cdots A_{j_1}$ is not an s.m.p.. Dominant s.m.p.s play a role for the termination of the invariant polytope algorithm discussed in Sections 1.1 and 4.

There are three common strategies to exploit Inequality (3): (i) Compute all products up to a length $k \in \mathbb{N}$ [16, 28]; (ii) Take a suitable family of norms and minimize the right hand side of (3) with respect to that family [1, 8, 9, 29]; (iii) Construct a norm which gives good estimates in (3) for short products, preferably for products of length one [17, 19–22, 26]. The Gripenberg algorithm [16], discussed in Section 3, belongs to class (i), the invariant polytope algorithm [17, 19–22], discussed in Section 1.1, to class (iii). The invariant polytope algorithm is, up to now, one of only two algorithms which can can compute the exact value of the JSR for a large number of matrix families. The second one is the infinite tree algorithm [27] which does not belong to any of the classes above. In this paper we concentrate on the invariant polytope algorithm.

We will call a norm $\|\cdot\|$ *extremal* for $\mathcal{A}$ if

$$\|A_j x\| \leq \mathrm{JSR}(\mathcal{A}) \cdot \|x\| \text{ for all } x \in \mathbb{R}^s \text{ and for all } A_j \in \mathcal{A}. \tag{4}$$

In [2] it is shown that every *irreducible* family of matrices, i.e. a family of matrices which have no trivial common invariant subspaces, possesses an extremal norm. Its construction is easily described in terms of the set

$$P(v) = \mathrm{co} \bigcup_{n \in \mathbb{N}_0,\ A_j \in \mathcal{A}} \left\{\pm A_{j_n} \cdots A_{j_1} v\right\}, \tag{5}$$

where co denotes the *convex hull* and $v \in \mathbb{R}^s$.

Theorem 1.1. *[3, 21]. If $\mathcal{A}$ is irreducible,* $\mathrm{JSR}(\mathcal{A}) \geq 1$ *and for a given $v \in \mathbb{R}^s$ the set $P(v)$ is bounded and has non-empty interior, then* $\mathrm{JSR}(\mathcal{A}) = 1$ *and $P(v)$ is the unit ball of an extremal norm $\|\cdot\|_{P(v)}$ for $\mathcal{A}$.*

Conversely, if $\mathcal{A}$ is irreducible and $\mathrm{JSR}(\mathcal{A}) = 1$*, then for any $v \in \mathbb{R}^s$, $P(v)$ is a bounded subset of $\mathbb{R}^s$.*

Clearly, the unit ball of a norm completely describes the corresponding norm. Given $P \subseteq \mathbb{R}^s$, a closed, convex and *balanced* ($\alpha P \subseteq P$ for all $|\alpha| < 1$) body with non-empty interior, the so-called *Minkowski norm* $\|\cdot\|_P : \mathbb{R}^s \to \mathbb{R}$,

$$\|\cdot\|_P = \inf\{r > 0 : x \in rP\} \tag{6}$$

fulfils $\{x \in \mathbb{R}^s : \|x\|_P \leq 1\} = P$.

The idea of the invariant polytope algorithm 1.4 is to construct an invariant set P for the matrices in the set $\mathcal{A}$ in finitely many steps. This is possible when P is a polytope.

We will describe polytopes by the convex hull of its vertices. For finite $V \subseteq \mathbb{R}^s$ we define the *symmetrized convex hull* of V by

$$\operatorname{co}_s V = \left\{x \in \mathbb{R}^s : x = \sum_{v \in V} t_v v \quad \text{with} \quad \sum_{v \in V} |t_v| \leq 1,\ t_v \in \mathbb{R}^s\right\} = \operatorname{co}(V \cup -V). \tag{7}$$

For finite $V \subseteq \mathbb{R}^s_+$ we define the *cone* of V with respect to the first orthant by

$$\operatorname{co}_- V = \left\{x \in \mathbb{R}^s_+ : x = y - z,\ y \in \operatorname{co}(V),\ z \in \mathbb{R}^s_+\right\}. \tag{8}$$

For simplicity, we denote with $\operatorname{co}_* V$ any of these convex hulls (co, co_s, co_-) depending on the context .

In all cases we identify a (finite) set V with the matrix whose columns are the coordinates of the points $v \in V$. Lemma 1.2 shows properties of Minkowski norms corresponding to various convex hulls.

Lemma 1.2. *Let $V \subseteq \mathbb{R}^s$ and $x \in \mathbb{R}^s$. Then*

(1) *If W are the vertices of another central symmetric polytope with non-empty interior such that $\operatorname{co}_* W \subseteq \operatorname{co}_* V$, then $\|\cdot\|_{\operatorname{co}_* V} \leq \|\cdot\|_{\operatorname{co}_* W}$.*
(2) *$\|x\|_{\operatorname{co}_* V} \leq \|t\|_1 \leq \sqrt{m}\|t\|_2$, where $Vt = x$, $t \in \mathbb{R}^m$.*
(3) *$\|x\|_{\operatorname{co}_s V} \geq \|V^+ x\|_2$, where V^+ is the Moore-Penrose pseudo-inverse of V.*
(4) *If there exists $w \in \mathbb{R}^s$ such that $|\langle w, v\rangle| < |\langle w, x\rangle|$ for all $v \in V$, then $x \notin \operatorname{co}_s V$.*
(5) *If $V \subseteq \mathbb{R}^s_+$, $x \in \mathbb{R}^s_+$ and there exists $v \in V$ such that $x_l \leq v_l$ for all $l = 1, \ldots, s$, then $x \in \operatorname{co}_- V$.*
(6) *If $V \subseteq \mathbb{R}^s_+$, $x \in \mathbb{R}^s_+$ and there exists $l \in \{1, \ldots, s\}$ such that $x_l > v_l$ for all $v \in V$, then $x \notin \operatorname{co}_- V$.*

Proof. (1) This immediately follows from the definition of the Minkowski norm.

(2) Let $x \in \operatorname{co}_s V$ and $t \in \mathbb{R}^{\#V}$ such that $x = Vt$. Define $\tilde{x} = \frac{x}{\|x\|_{\operatorname{co}_s V}} \in \partial \operatorname{co}_s V$ and $\tilde{t} = \frac{t}{\|x\|_{\operatorname{co}_s V}}$. It follows that $\tilde{x} = V\tilde{t}$ with $\|\tilde{t}\|_1 \geq 1$. Indeed, $\|\tilde{t}\|_1 < 1$ would imply that $\tilde{x} \in (\operatorname{co}_s V)^\circ$. Clearly, $1 = \|\tilde{x}\|_{\operatorname{co}_s V} \leq \|\tilde{t}\|_1$ and $\|x\|_{\operatorname{co}_s V} \leq \|t\|_1$. Finally, by (1), $\|x\|_{\operatorname{co}_- V} \leq \|x\|_{\operatorname{co}_s V}$. The second inequality follows from the equivalence of norms.

(3) Let $x \in \operatorname{co}_s V$ and $t \in \mathbb{R}^{\#V}$ such that $x = Vt$. It follows that $\|t\|_1 \geq \|t\|_2 \geq \|V^+ x\|_2$ because, by construction of the Moore-Penrose pseudo inverse, $V^+ x$ is the unique solution to $Vt = x$ with minimum 2-norm. Finally, by (2), $\|x\|_{\operatorname{co}_s V} = \min_{t \in \mathbb{R}^{\#V} : Vt = x} \|t\|_1 \geq \|V^+ x\|_2$.

(4) If $|\langle w, v\rangle| < |\langle w, x\rangle|$ for all $v \in V$, then there exists a hyperplane which separates the point x and the polytope $\operatorname{co}_s V$. From this the claim directly follows.

(5) Defining $z := v - x$ we see that $z \in \mathbb{R}^s_+$ which implies $x = v - z \in \operatorname{co}_- V$.

(6) Since $\operatorname{co}_s V$ is convex, $y_l \leq v_l$ for all $y \in \operatorname{co} V \cap \mathbb{R}^s_+$. Since $z_l > 0$ it follows that $y_l - z_l \leq y_l \leq v_l < x_l$. Thus, there does not exist y, z such that $x = y - z$.

□

Remark 1.3. Estimate 1.2 (5) uses the fact that the norms $\|\cdot\|_{\operatorname{co}_- V}$ are orthant monotonic, i.e. $\|x\|_{\operatorname{co}_- V} \leq \|y\|_{\operatorname{co}_- V}$, $x, y \in \mathbb{R}^s_+$ whenever $0 \leq x_i \leq y_i$ for all $i = 1, \ldots, s$. It would be interesting to know whether and when Minkowski norms $\|\cdot\|_{\operatorname{co}_s V}$ or Minkowski norms composed with linear mappings $\|M \cdot\|_{\operatorname{co}_s V}$, $M \in \mathbb{R}^{s \times s}$, are orthant monotonic. This would allow to transfer the estimate 1.2 (5) to Minkowski norms corresponding to symmetrised convex hulls $\operatorname{co}_s V$.

1.1 Invariant polytope algorithm and outline for the paper

In this section we present the idea of the invariant polytope algorithm. The major topic of this paper are modifications to the invariant polytope algorithm making it

- faster and parallel,
- more robust and
- more efficient for larger matrices.

These modifications are outlined in Section 2. The actual modified invariant polytope algorithm 4.1 is given in Section 4. In Section 3 we introduce the modified Gripenberg algorithm 3.1 which is capable of finding very long s.m.p.-candidates in short time. Section 5 is devoted to numerical examples showing where the algorithms from Sections 3 and 4 perform well and where they are not applicable any more.

Algorithm 1.4 (Invariant polytope algorithm [17, 19–22]). Given $\mathcal{A} = \{A_j : j = 1, \dots, J\} \subseteq \mathbb{R}^{s\times s}$.

(1) For some $D \in \mathbb{N}$ look over all products of matrices in $\mathcal{A}$ of length less than D and choose a shortest product Π_1 such that $\rho_c := \rho(\Pi_1)^{1/l_1}$ is maximal, where l_1 is the length of the product and call Π_1 *spectral maximizing product-candidate* (*s.m.p.-candidate*). Set $\tilde{\mathcal{A}} := \rho_c^{-1}\mathcal{A}$. Now we try to prove that $\mathrm{JSR}(\tilde{\mathcal{A}}) \le 1$.
(2) Let v_1 be the *leading eigenvector* of Π_1 i.e. the eigenvector with respect to the largest eigenvalue in magnitude
(3) Construct the cyclic root $\mathcal{H}$: Let $v_1^{(i)}$, $i = 1, \dots, l_1 - 1$, be the leading eigenvectors of the cyclic permutations of $\tilde{\Pi}_1$, i.e. for $\tilde{\Pi}_1 = \tilde{A}_{j_{l_1}} \cdots \tilde{A}_{j_1}$ we get $v_1^{(i)} := \tilde{A}_{j_i} \cdots \tilde{A}_{j_1} v_1$. Set $\mathcal{H} := \{v_1^{(0)}, \dots, v_1^{(l_1-1)}\}$ and $V := \mathcal{H}$.
(4) For all $v \in V$ and for all $j = 1, \dots, J$
    If $\|\tilde{A}_j v\|_{\mathrm{co}_* V} > 1$ set $V := V \cup \tilde{A}_j v$.
    Depending on $\mathcal{A}$ and the leading eigenvector v_0 we use different convex hulls:
    case (P): If all entries of the matrices A_j are non-negative, then we can take a non-negative leading eigenvector v_0 in step (2) and use co_-.
    case (R): If the matrices A_j have positive and negative entries and the leading eigenvector v_0 is real, then we use co_s;
(5) Repeat step (4) until $\tilde{\mathcal{A}}V \subseteq \mathrm{co}_* V$.
(6) If $\tilde{\mathcal{A}}V \subseteq \mathrm{co}_* V$, then the algorithm terminates and we have found an invariant polytope $\mathrm{co}_* V$, which implies that $\|\tilde{A}_j\|_{\mathrm{co}_* V} \le 1$ for all $j = 1, \dots, J$, or in other words, $\mathrm{JSR}(\tilde{\mathcal{A}}) \le 1$.

Remark 1.5. In step 1.4 (4) we actually add a vertex $\tilde{A}_j v \notin \mathcal{H}$ even if it lies slightly inside of the polytope, i.e. if $\|\tilde{A}_j v\|_{\mathrm{co}_* V} > 1 - \epsilon$, where $\epsilon > 0$ is the accuracy up to which the norm can be computed. This is important to obtain a mathematically rigorous result.

Remark 1.6. If neither *case* (P) nor *case* (R) applies, i.e. the matrices are not strictly non-negative and have complex leading eigenvalues, then one would have to consider complex polytopes, which is not discussed in this paper.

Figure 1 presents the invariant polytope algorithm on some concrete example.

2 SUMMARY OF THE MAIN MODIFICATIONS

In this section we present the modifications to the invariant polytope algorithm 1.4 and explain their importance. For more details see Sections 3 and 4.

2.1 New balancing procedure

In steps (2) and (3) of the explanation of the invariant polytope algorithm in 1.4, we only had one cyclic root, corresponding to the one leading eigenvector v_1. If there happens to be more than one cyclic root, then it is necessary to balance the sizes of the cyclic roots to each other in order to ensure termination of the invariant polytope algorithm [19]. There are (at the moment) three reasons why

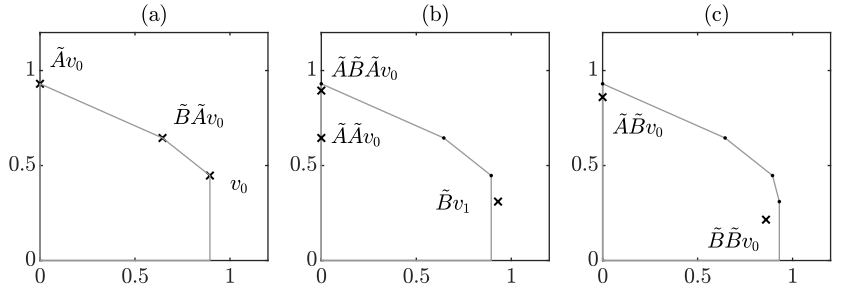

Fig. 1. The polytope $\mathrm{co}_- V$ as constructed by the invariant polytope algorithm 1.4 for the matrices $A = \left[\begin{smallmatrix} 0 & 0 \\ 1 & 1 \end{smallmatrix}\right]$, $B = \left[\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right]$, In (a) we see the cone $\mathrm{co}_- \mathcal{H}$ with respect to the cyclic root $\mathcal{H} = \{v_1, \tilde{A}v_1, \tilde{B}\tilde{A}v_1\}$. In (b) we see the vertices $\tilde{A}v_1$, $\tilde{A}\tilde{A}v_1$ and $\tilde{A}\tilde{B}\tilde{A}v_1$ constructed in the first iteration. In (c) we see the new polytope $\mathrm{co}_-(\mathcal{H} \cup \tilde{B}v_1)$ together with the vertices $\tilde{B}\tilde{B}v_1$ and $\tilde{A}\tilde{B}v_1$ constructed in the second iteration, which are all mapped into the interior of $\mathrm{co}_- \mathcal{H} \cup \tilde{B}v_1$.More precise:
All entries of A and B are non-negative, thus, we are in case (P) and use the cone hull co_- to compute the Minkowski-norms in step 1.4 4.
(1) We choose $\Pi_1 = BBA$, which is the product with the highest averaged spectral radius among all products of length less or equal than three. Thus, $l_1 = 3$, $\rho_c = \rho(\Pi_1)^{1/l_1} = 3^{1/3}$ and we define $\tilde{A} = \rho_c^{-1}A$, $\tilde{B} = \rho_c^{-1}B$, $\tilde{\mathcal{A}} = \{\tilde{A}, \tilde{B}\}$, $\tilde{\Pi}_1 = \tilde{B}\tilde{B}\tilde{A}$.
(2) The s.m.p.-candidate $\tilde{\Pi}_1$ has only one simple leading eigenvalue 1 with a corresponding eigenvector $v_1 = v_1^{(0)}$ given by $v_1 = v_1^{(0)} = 5^{-1/2}[\,2 \quad 1\,]^T$.
(3) We construct the cyclic root $\mathcal{H} = \{v_1^{(0)}, v_1^{(1)}, v_1^{(2)}\} = \{v_1, \tilde{A}v_1, \tilde{B}\tilde{A}v_1\} = 5^{-1/2}\{[\,1 \quad 2\,]^T, [\,0 \quad 3^{2/3}\,]^T, [\,3^{1/3} \quad 3^{1/3}\,]^T\}$ and set $V = \mathcal{H}$.
(4, first iteration) We compute the norms of the vectors $\tilde{\mathcal{A}}V \setminus V$. The vector $\tilde{B}v_1$ is outside of the polytope $\mathrm{co}_- V$, $\|\tilde{B}v_1\|_{\mathrm{co}_- V} \simeq 1.04 \geq 1$, and thus, it is added to the set V. All other vectors, i.e. $\tilde{A}\tilde{A}v_1$ and $\tilde{A}\tilde{B}\tilde{A}v_1$, in the first iteration are inside of $\mathrm{co}_- V \cup \tilde{B}v_1$; $\|\tilde{A}\tilde{A}v_1\|_{\mathrm{co}_- V \cup \tilde{B}v_1} \simeq 0.69 < 1$, $\|\tilde{A}\tilde{B}\tilde{A}v_1\|_{\mathrm{co}_- V \cup \tilde{B}v_1} \simeq 0.96 < 1$.
(4, second iteration) We repeat step 4 and test the vectors from the set $\tilde{\mathcal{A}}(V \cup \tilde{B}v_1) \setminus (V \cup \tilde{B}v_1)$; $\|\tilde{B}\tilde{B}v_1\|_{\mathrm{co}_- V \cup \tilde{B}v_1} \simeq 0.92 < 1$, $\|\tilde{A}\tilde{B}v_1\|_{\mathrm{co}_- V \cup \tilde{B}v_1} \simeq 0.92 < 1$.
(5) All vertices from the second iteration are mapped into the interior of the polytope $P = \mathrm{co}_- V \cup \tilde{B}v_1$, therefore, P is $\tilde{\mathcal{A}}$-invariant and $\mathrm{JSR}(\mathcal{A}) = \rho(\Pi_1)^{1/l_1} = 3^{1/3} \simeq 1.4422$.

multiple cyclic roots occur: (1) The s.m.p.-candidate Π_1 possesses more than one leading eigenvalue, or its leading eigenvalue is not simple, or there are more than one s.m.p.-candidates $\Pi_1 \ldots, \Pi_R$, $R \in \mathbb{N}$. But, multiple cyclic roots also can be generated by (2) artificially adding cyclic roots or by (3) artificially adding individual vertices. Technique (2) usually is employed whenever there are matrix products whose averaged spectral radius is *nearly* that of the s.m.p.-candidate. Such matrix products are usually called *nearly-s.m.p.s* [19, Remark 3.7]. If the leading eigenvectors of a nearly-s.m.p. are complex, one can take a real pair of vectors spanning the space generated by the complex eigenvectors. Technique (3) usually is employed whenever the initial polytope $\mathrm{co}_* \mathcal{H}$ is very flat [19, Section 4].

The original balancing procedure described in [19, Sections 2.3 and 3] may fail for multiple cyclic roots caused by the presence of nearly-s.m.p.s.. In Section 4.5 we improve on the original implementation such that it always works and, in addition, automatically. In Section 4.4 we suggest an automated procedure how to select good nearly-s.m.p.s and extra vertices. Aside from that, Example 4.3 presents a set of matrices where it was wrongly assumed that no balancing is necessary.

2.2 Finding s.m.p. candidates

The invariant polytope algorithm 1.4 only terminates if all s.m.p.-candidates Π_r, $r = 1, \dots, R$, are indeed s.m.p.s.. Thus, the invariant polytope algorithm 1.4 heavily relies on correct initial guesses for the s.m.p.-candidates. A plain brute-force search in 1.4 (1) will fail, if the s.m.p.s length is large. Our numerical tests have shown that even for random pairs of 2×2 matrices s.m.p.s of length greater than 30 are not uncommon. A particular easy example is given in Example 5.2. We present two new methods that search for s.m.p.s efficiently in Sections 3 and 4.10.

2.3 Bounds for the JSR

If the invariant polytope algorithm 1.4 does not find an invariant polytope in reasonable time, it can still give an upper bound for the JSR *after* termination. In Lemma 4.2 we show that our modified invariant polytope algorithm can return bounds for the JSR in *each* iteration of the modified invariant polytope algorithm without the need of terminating the algorithm.

Nevertheless, these bounds are usually quite rough. A simple modification, presented in Remark 4.3, increases the accuracy of these intermediate bounds on the drawback that the exact value of the JSR becomes incomputable.

2.4 Parallelization and natural selection of vertices

A disadvantage of the invariant polytope algorithm 1.4 in its current form is that the polytope is changed *inside* of the main loop in 1.4 (4), which implies that in general the norm of $\tilde{A}_j v$ has to be computed with respect to a different polytope for each vertex. Therefore, the linear programming problem is different for each norm and the so-called warm start of linear programming problems cannot be used. Furthermore, the main loop cannot be parallelised. We eliminate these two drawbacks and additionally speed up the invariant polytope algorithm in Section 4.8.

The employed technique also solves a problem arising when the number of matrices in $\mathcal{A}$ is large. In such cases the invariant polytope algorithm 1.4 will stall, simply due to the fact, that the number of vertices to test, increases in the worst case by a factor of $\#\mathcal{A}$ in each iteration. E.g., if $\#\mathcal{A} \gtrsim 100$, the original invariant polytope algorithm is likely never to reach the third iteration.

2.5 Estimating the Minkowski norm

To reduce the number of norms one has to compute in 1.4 (4), we use the estimates for the Minkowski norm in Lemma 1.2.

3 MODIFIED GRIPENBERG ALGORITHM

From Inequality (3) we know that the normalized spectral radius of any matrix product is a lower bound for the JSR. Thus, by a clever guess of a matrix product one easily obtains good (maybe sharp) lower bounds for the JSR. Our new modified Gripenberg algorithm presented in this section finds in nearly all of our numerical tests an s.m.p..

The modified Gripenberg algorithm 3.1 is a modification of the well-known Gripenberg algorithm [16], one of the first algorithms which gave reasonable estimates for the JSR. We briefly describe how it works: Given some accuracy $0 < \delta \leq 1$ we iteratively compute the sets C_k, $k \in \mathbb{N}_0$. $C_0 := I$ and C_{k+1} consists of all matrices $C \in \mathcal{A}C_k$ with $\|C\|^{1/(k+1)} \geq \delta^{-1} b_-$, where

$$b_- = \max\{\rho(C)^{1/n} : C \in C_n,\ n = 1, \dots, k\}$$

is the current lower bound for the JSR. In other words, we sort out matrix products whose averaged norm is less than the current lower b_- bound of the JSR. For each k the JSR lies in the interval

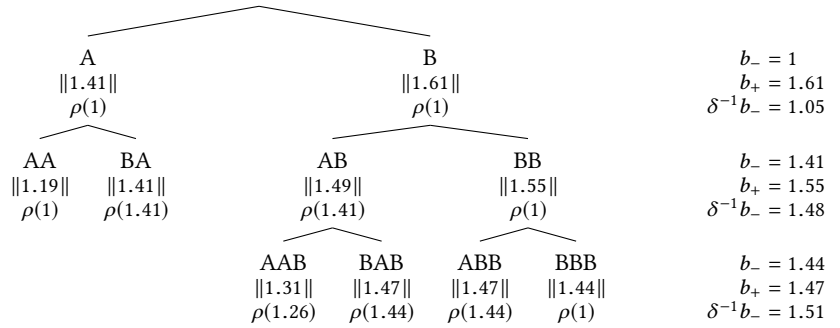

Fig. 2. Tree built up by Gripenberg's algorithm for the matrices $A = \left[\begin{smallmatrix} 0 & 0 \\ 1 & 1 \end{smallmatrix}\right]$ and $B = \left[\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right]$ with $\delta = 0.95$ and using the 2-norm. The computed matrices, their averaged norms and averaged spectral radii are printed. Gripenberg's algorithm terminates after the third iteration, since all extant matrices have averaged norm less then $\delta^{-1}b_-$. Thus, JSR $\in [1.44,\ 1.47]$. More precise:
Iteration 1 Gripenberg's algorithm starts computing (averaged) norms and spectral radii of the matrices in the set $C_1 = \mathcal{A} \cdot \{I\} = \{A, B\}$; $\|A\|_2 \simeq 1.41$, $\|B\|_2 \simeq 1.61$, $\rho(A) = 1$, $\rho(B) = 1$. Thus, we get the lower and upper bounds $b_- = \max\{\rho(A), \rho(B)\} = 1$ and $b_+ = \max\{\|A\|_2, \|B\|_2\} \simeq 1.61$ for the JSR. The norms of both matrices is larger than $\delta^{-1}b_-$, thus, $C_2 = \mathcal{A}\{A, B\} = \{AA, BA, AB, BB\}$.
Iteration 2 Computing all averaged norms and spectral radii from the matrices in the set C_2, we obtain $b_- \simeq 1.41$, $b_+ \simeq 1.55$. Since, $\|AA\|_2^{1/2}, \|BA\|_2^{1/2} < \delta^{-1}b_-$ we define $C_3 = \mathcal{A}\{AB, BB\}$.
Iteration 3 Computing all averaged norms and spectral radii from the matrices in the set C_3, we obtain $b_- \simeq 1.44$, $b_+ \simeq 1.45$. The averaged norms of all matrices in the set C_3 is less than b_-, and thus, $C_4 = \emptyset$. The algorithm terminates and returns JSR$(\mathcal{A}) \,\widetilde{\in}\, [1.44, 1.47]$. Note that, indeed, $\delta \leq 0.98 \simeq 1.44/1.47$.

$[b_-,\ b_+]$ with

$$b_+ = \min_{n=1,\dots k} \max\left\{\|C\|^{1/n} : C \in C_n\right\}.$$

Note that b_- is monotone increasing and b_+ is monotone decreasing. If $\delta < 1$ Gripenberg's algorithm terminates [16], i.e. there exists $K \in \mathbb{N}$ such that $C_K = \emptyset$ and the JSR is computed up to an accuracy of δ, i.e. $b_-/b_+ \leq \delta$. For real-world applications Gripenberg's algorithm works well for $\delta \leq 0.95$. For larger δ the number of products to compute is usually too large. Figure 2 shows how to estimate the JSR using Gripenberg's algorithm for a concrete set of matrices. For some vertex $w = A_j$, $v \in V_k$, $k \in \mathbb{N}$, we say that w is a *child* of v, and that v is the *parent* of w.

The modified Gripenberg algorithm 3.1 uses a different mechanism to sort out matrix products. Instead of just dismissing products with norms less than some threshold, it furthermore only keeps products with highest and lowest norms,

Algorithm 3.1 (modified Gripenberg algorithm).

Input :

Set of square matrices $\mathcal{A} = \{A_j : j = 1, \dots, J\} \subseteq \mathbb{R}^{s\times s}$

Number of products kept in each step $N \in \mathbb{N}$

Maximal length of products $D \in \mathbb{N}$

Output :

S.m.p.-candidates C

Lower bound ρ_c for JSR$(\mathcal{A})$

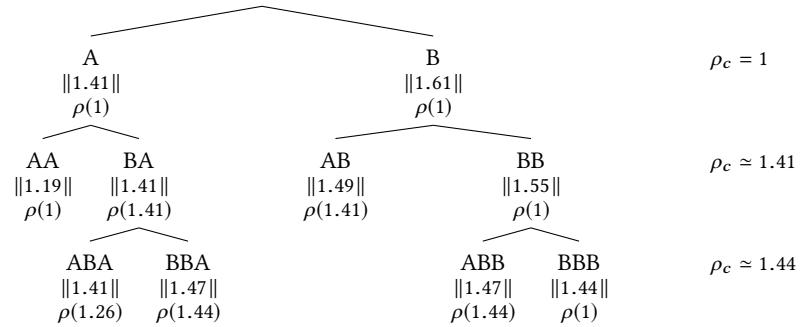

Fig. 3. Tree built up by the modified Gripenberg algorithm for the matrices $A = \left[\begin{smallmatrix} 0 & 0 \\ 1 & 1 \end{smallmatrix}\right]$ and $B = \left[\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right]$ with $N = 1$, $D = 3$ and using the 2-norm. The computed matrices, their averaged norms and averaged spectral radii are printed. The modified Gripenberg algorithm returns that $\mathrm{JSR}(\{A, B\}) \gtrsim 1.44$. More precise:
Iteration 1 We set $\mathcal{M}_0 = \{I\}$, $\rho_c = 0$. The modified Gripenberg algorithm starts computing averaged norms and spectral radii of the matrices in the set $\mathcal{M}_1 = \mathcal{A}\mathcal{M}_0 = \{A, B\}$; $\|A\|_2 \simeq 1.41$, $\|B\|_2 \simeq 1.61$, $\rho(A) = 1$, $\rho(B) = 1$. Thus, $\rho_c = \max\{0, \rho(A), \rho(B)\} = 1$. Since $\|A\|_2, \|B\|_2 \geq \rho_c$ no matrix products are removed, and $\mathcal{M}_1 = \{A, B\}$. After sorting with respect to the (averaged) norms we obtain $\mathcal{M}_1 = \{B, A\}$. Since $N = 1$ we keep the first and last element of the sorted set, thus, $\mathcal{M}_1 = \{B, A\}$.
Iteration 2 Computing the averaged norms and spectral radii in the set $\mathcal{M}_2 = \mathcal{A}\mathcal{M}_1$. we obtain $\rho_c \simeq 1.41$. Since $\|AA\|_2^{1/2} < \rho_c$ we set $\mathcal{M}_2 = \{BA, AB, BB\}$. After sorting with respect to the averaged norms we obtain $\mathcal{M}_2 = \{BB, AB, BA\}$ and since $N = 1$ we keep the first and last element,thus, $\mathcal{M}_2 = \{BB, BA\}$.
Iteration 3 Computing the averaged norms and spectral radii in the set $\mathcal{M}_3 = \mathcal{A}\mathcal{M}_2$. we obtain $\rho_c \simeq 1.44$. Since $D = 3$ we stop in this iteration and return $\mathrm{JSR}(\mathcal{A}) \gtrsim 1.44$ and the set of s.m.p.-candidates $C = \{BBA\}$.

Initialization :

Start with the product of length 0, $\mathcal{M}_0 := \{I\}$, where I is the identity matrix

Set current lower bound for JSR, $\rho_c := 0$

Algorithm :

for $d = 1, \ldots, D$

Compute all possible new products $\mathcal{M}_d := \mathcal{A}\mathcal{M}_{d-1}$

Update lower bound $\rho_c := \max\{\rho_c,\ \rho(M_d)^{1/d} : M_d \in \mathcal{M}_d\}$

Remove products whose norms are less than ρ_c, $\mathcal{M}_d := \{M_d \in \mathcal{M}_d : \|M_d\|^{1/d} \geq \rho_c\}$

Keep only products with highest and lowest norms:

Sort $\mathcal{M}_d$ w.r.t $\|M_d\|$ and sort out matrices with indices $N + 1, \ldots, \#\mathcal{M}_d - N - 1$ (9)

Thus $\mathcal{M}_d = \{M_1, \ldots, M_N, M_{\#\mathcal{M}_d - N}, \ldots, M_{\#\mathcal{M}_d} : M_i \in \mathcal{M}_d\}$

Post processing :

Choose products $C = \{M_{d_i} \in \mathcal{M}_d : \rho(M_{d_i})^{1/d} = \rho_c,\ d = 1, \ldots, D\}$

Remove cyclic permutations and powers of products from C

return C, ρ_c

Theorem 3.2. *The modified Gripenberg algorithm 3.1 has linear complexity in the number $J = \#\mathcal{A}$ of matrices , in the number $N \in \mathbb{N}$ of kept products in each level and in the maximal length $D \in \mathbb{N}$ of the products.*

Proof. In every iteration, in total D many, the modified Gripenberg algorithm computes at most $2 \cdot N \cdot J$ norms and spectral radii. □

Remark 3.3. The modified Gripenberg algorithm 3.1 in the given form only returns lower bounds for the JSR. If one keeps track which products are dismissed, then it is possible to give also upper bounds for the JSR. Note that the modified Gripenberg algorithm with parameters $N = D = \infty$ is exactly Gripenberg's algorithm with accuracy $\delta = 1$,

Remark 3.4. Clearly one can pursue other selection strategies in step (9). The straightforward choice of taking the $2 \cdot N$ products with highest normalized norm performs very badly. Taking an arbitrary subset of $\mathcal{M}_d$ of size $2 \cdot N$ in step 3.1 (9) performs mostly similarly to the modified Gripenberg algorithm 3.1, but in some cases worse, see Table 3 where we call it *random Gripenberg algorithm*. Furthermore, the modified Gripenberg algorithm 3.1 in the given form is deterministic, so we prefer it over a non-deterministic version.

Remark 3.5. Our new modified invariant polytope algorithm, presented in Section 4, can also be used to search for s.m.p.-candidates. Thus, we give the numerical examples showing the performance of the modified Gripenberg algorithm 3.1 only after Section 4.

Figure 3 shows how to find lower bounds for the JSR using the modified Gripenberg algorithm for a concrete set of matrices. You may want to compare this Figure with Figure 2.

4 MODIFIED INVARIANT POLYTOPE ALGORITHM

In this section, we present the modifications to the invariant polytope algorithm 1.4.

Algorithm 4.1 (Modified invariant polytope algorithm). Lines with numbers are subroutines, described in detail in Sections 4.1–4.10.

Input :

Set of irreducible square matrices $\mathcal{A} = \{A_j : j = 1, \dots, J\} \subseteq \mathbb{R}^{s \times s}$ (10)

Accuracy $0 < \delta \leq 1 \quad (\delta \simeq 1)$

Accuracy $0 < \epsilon < 1$ for computing the norms $N(v)$ in (17) $(\epsilon \simeq 0)$

Output :

Exact value ρ_c or bound $[\rho_c,\ b \cdot \rho_c]$ for JSR($\mathcal{A}$)

Invariant polytope $\operatorname{co}_* V$

Spectral maximizing products Π_r

Initialization :

Search for s.m.p.-candidates and nearly-s.m.p.s $\Pi_r = A_{j_{r_{l_r}}} \cdots A_{j_{r_1}},\ r = 1, \dots, R$ (11)

Set $\rho_r := \rho(\Pi_r)^{1/l_r},\ \rho_c := \max \rho_r,\ \tilde{\mathcal{A}} := \delta \rho_c^{-1} \mathcal{A}$ (12)

Compute the leading eigenvectors v_r of $\tilde{\Pi}_r$

Compute the root vectors $v_r^{(i)} := (\rho_c / \delta \rho_r)^i \tilde{A}_{j_{r_i}} \dots \tilde{A}_{j_{r_1}} v_r,\ i = 0, \dots, l_r - 1$

Compute the extra-vertices $v_{R+1}, \dots, v_S \in \mathbb{R}^s$ (13)

Compute the balancing factors $\alpha_1, \dots, \alpha_S \in \mathbb{R}$ (14)

Set $\mathcal{H} := \{\alpha_1 v_1^{(0)}, \alpha_1 v_1^{(1)}, \alpha_1 v_1^{(2)}, \dots, \alpha_R v_R^{(l_R - 1)}\},\ V_0 := \mathcal{H} \cup \{\alpha_{R+1} v_{R+1}, \dots, \alpha_S v_S\}$

Set $N(v) := \infty$ for all $v \in V_0$, $\quad b_0 := \infty$, $\quad k := 0$

Main Loop :

while $\tilde{\mathcal{A}} V_k \setminus \mathcal{V}_k \not\subseteq (1-\epsilon)\,\mathrm{co}_*\, V_k$

Select new children $E_{k+1} \subseteq \tilde{\mathcal{A}} V_k \setminus \mathcal{V}_k$ based on norm estimates (15)

Choose subset of vertices $W_k \subseteq V_k$ (16)

Compute/classify norm $N(v) := \|v\|_{\mathrm{co}_* W_k}$ for all $v \in E_{k+1}$ (17)

$V_{k+1} := V_k \cup \{v \in E_{k+1} : N(v) > 1-\epsilon\}$ (18)

$b_{k+1} := \min\left\{b_k,\ \max\{1,\ N(v)(1-\epsilon)^{-1} : v \in V_{k+1} \wedge \tilde{\mathcal{A}} v \not\subseteq V_{k+1}\}\right\}$

Test spectral radii based and eigenplane based stopping critera (19)

print JSR $\in [\rho_c,\ \delta^{-1} \cdot b_{k+1} \cdot \rho_c]$

$k := k+1$

return $V,\ \{\Pi_r\}_r,\ \rho_c$

Theorem 4.2. *Let $\mathcal{A} = \{A_j : j = 1, \ldots, J\} \subseteq \mathbb{R}^{s\times s}$ be a finite set of square matrices.*

(i) For $\delta = 1$, the modified invariant polytope algorithm 4.1 terminates if and only if the original invariant polytope algorithm 1.4 terminates, i.e. $\Pi_1, \ldots, \Pi_R$ are dominant s.m.p.s and each s.m.p. possesses only one simple leading eigenvalue[1].

(ii) For $0 < \delta < 1$ the modified invariant polytope algorithm 4.1 terminates if $\mathrm{JSR}(\mathcal{A}) < \delta^{-1} \cdot \rho_c$.

(iii) Moreover, for any iteration $k \in \mathbb{N}_0$, $\mathrm{JSR}(\mathcal{A}) \in [\rho_c,\ \delta^{-1} \cdot b_{k+1} \cdot \rho_c]$, where ρ_c and and b_{k+1} are defined in Algorithm 4.1.

Before presenting the proof of Theorem 4.2 in Section 4.11, we describe all modifications and extensions to the original invariant polytope algorithm 1.4. These are numbered (10)–(19) in the modified invariant polytope algorithm 4.1. All heuristic constants which influence the behaviour of the algorithm can be changed by passing a name-value pair in the function call of our implementation, see the documentation for more information.

4.1 Irreducibility of input matrices (10)

The set of matrices $\mathcal{A}$ should be irreducible, i.e. the matrices in the set $\mathcal{A}$ should not have a trivial common invariant subspace, because otherwise (both the modified 4.1 and) the invariant polytope algorithm 1.4 may not be able to terminate. If the matrices are reducible, then there exists a basis in which all of the matrices A_j have block upper triangular form. The JSR of the matrices then equals to the maximum of the JSR of the diagonal blocks. In our implementation we therefore automatically search for non-trivial common invariant subspaces prior to starting the modified invariant polytope algorithm. Here we make use of the functions `permTriangul` and `jointTriangul` from [25], as well as a new method `invariantsubspace` which searches for non-trivial common invariant difference subspaces as described in [13].

4.2 Search for s.m.p.-candidates (11)

We use the modified Gripenberg algorithm 3.1 to search for s.m.p.-candidates and nearly-s.m.p.s. Every product, which is shorter than the s.m.p.-candidate and having normalized spectral radius greater or equal to $\tau \cdot \rho_c$ is considered to be a nearly-s.m.p.. In our implementation we use a heuristic default value of $\tau = 0.9999$ and use the Matlab function `eig` to compute the leading eigenvalues.

[1] In [17] such eigenvalues are called *unique*.

This may not be the fastest available procedure, but it is fast enough in comparison to the time the main loop needs to terminate.

4.3 Approximate computation (12)

If we multiply the set of matrices $\tilde{\mathcal{A}}$ by a factor $0 < \delta < 1$, the modified invariant polytope algorithm 4.1 cannot return exact values for the JSR anymore, but only up to a relative accuracy of δ. Indeed, if the modified invariant polytope algorithm 4.1 terminates, then $\|\tilde{A}_j v\|_{\mathrm{co}_* V} \leq 1 \Leftrightarrow \|A_j v\|_{\mathrm{co}_* V} \leq \delta^{-1} \cdot \rho_c \Leftrightarrow \mathrm{JSR}(\mathcal{A}) \leq \delta^{-1} \cdot \rho_c$. There are cases where this procedure is of significance.

(a). If the dimension s of matrices is large, (both the modified 4.1 and) the invariant polytope algorithm 1.4, will probably not terminate anyway, and thus only give bounds for the JSR. A factor $\delta \simeq 0.97$ will speed up the computation tremendously and the returned bounds from the modified invariant polytope algorithm are mostly better (at least in our numerical examples) than for $\delta = 1$. The value 0.97 is based on numerical experiments. An optimal value for δ can probably be determined using the spectral gap at 1, but no theoretical investigations nor numerical experiments in that direction have been taken so far.

(b). If the s.m.p.s are not dominant, or there is an infinite number of dominant s.m.p.s, or $\mathcal{A}$ is not irreducible, the modified invariant polytope algorithm 4.1 will not terminate. In these cases, choosing $\delta \simeq 1 - 10^{-9}$ ensures that the modified invariant polytope algorithm 4.1 terminates and the obtained bounds will be nearly the same as when $\delta = 1$. Note that these cases are mostly non-generic, except for matrix families where this property is known to hold a priori, for example certain matrix sets occurring in subdivision.

(c). If one is interested only whether $\mathrm{JSR}(\mathcal{A}) < B$ for some $B > 0$, one can choose $1 > \delta > B^{-1}\rho_c$ and the modified invariant polytope algorithm 4.1 will terminate much faster.

4.4 Adding extra-vertices automatically (13)

The aim of this step in the algorithm is, to compute vertices $E = \{v_{R+1}, \ldots, v_S\}$ such that the polytope $\mathrm{co}_*(\mathcal{H} \cup E)$, $\mathcal{H} = [v_1^{(0)}, v_1^{(1)}, \ldots, v_R^{(l_R-1)}]$, has non-empty interior and is elongated in all coordinate directions. The procedure is different in cases (P) and (R).

For case (R), given some threshold $T > 0$, we compute the singular value decomposition of $\mathcal{H} = [v_1^{(0)}, v_1^{(1)}, \ldots, v_R^{(l_R-1)}]$. and take all singular vectors (which thus become *extra-vertices*) $E = \{v_{R+1}, \ldots, v_S\}$ corresponding to singular values which are in modulus less than T. Note that the singular vectors form an orthonormal system and that the singular vectors corresponding to small singular values are exactly the directions in which the polytope $\mathrm{co}_* \mathcal{H}$ has small or even no elongation. In particular, the polytope $\mathrm{co}_*(\mathcal{H} \cup E)$ has always non-empty interior.

For case (P), $e_n \in E$ whenever $v_n \leq T$ for all $v \in \mathcal{H}$, where e_n is the n^{th} unit vector of $\mathbb{R}^s$,

In our implementation we use a heuristic value of $T \simeq 0.1$ for both cases.

4.5 Balancing of cyclic trees (14)

As already noted, the existence of multiple cyclic roots makes it necessary to balance the sizes of the cyclic roots to each other in order that the invariant polytope algorithm can terminate. The balancing procedure uses the *dual leading eigenvectors* v_r^*, $r = 1, \ldots, R$. More precisely, for the s.m.p.-candidate $\tilde{\Pi}_r$ define $\tilde{\Pi}_r^* v_r^* = v_r^*$ with $\langle v_r^{(0)}, v_r^* \rangle = 1$, where Π_r^* is the conjugate transpose of Π_r and $\langle \cdot, \cdot \rangle$ is the standard inner product [19, Section 2.3].

If $\delta < 1$ no balancing is necessary by Theorem 4.2. If $\delta = 1$ we define for $h \in \mathbb{N}$

$$\begin{cases} q_{i,j} = \sup\limits_{z \in \tilde{\mathcal{A}}^h \{(\rho_c/\rho_r)^i v_i^{(0)}, \ldots, v_i^{(l_i-1)}\}} |\langle v_j^*, z\rangle|, & i = 1, \ldots, R \\ q_{i,j} = \sup\limits_{z \in \tilde{\mathcal{A}}^h v_i} |\langle v_j^*, z\rangle|, & i = R+1, \ldots, S \end{cases}, \ j = 1, \ldots, R.$$

The factor $(\rho_c/\rho_r)^i$ ensures that all vertices of the cyclic root of nearly-s.m.p.s get the same weight in the computation. If v_i is the leading eigenvector of an s.m.p.-candidate, we have $\rho_c/\rho_r = 1$. Now one has to find numbers $\alpha_1, \ldots, \alpha_S > 0$ such that

$$\begin{cases} \alpha_i q_{i,j} < \alpha_j & \text{whenever } v_i \text{ is the leading eigenvector of an s.m.p.-candidate} \\ \alpha_i q_{i,j} < 1 & \text{otherwise} \end{cases}$$

and multiply all vertices $v_i^{(j)}$, $i = 1, \ldots, R$, $j(i) = 0, \ldots, l_i - 1$, and extra-vertices v_i, $i = R+1, \ldots, S$, from the root $\mathcal{H}$ with the corresponding balancing factor α_i. In our implementation we distinguish between extra-vertices and vertices from nearly-s.m.p.s., precisely we solve the following system

$$\begin{cases} \alpha_i q_{i,j} < \alpha_j & \text{whenever } v_i \text{ is the leading eigenvector of an s.m.p.-candidate} \\ \alpha_i q_{i,j} = B_{\text{nearly}} \cdot \rho_i & \text{whenever } v_i \text{ is the leading eigenvector of a nearly-s.m.p.} \\ \alpha_i q_{i,j} = B_{\text{extra}} & \text{whenever } v_i \text{ is an extra-vertex} \end{cases},$$

where $B_{\text{nearly}} = 0.999$ and $B_{\text{extra}} = 0.01$ are based on numerical experiments. [19, Theorem 3.3] ensures that the modified invariant polytope algorithm 4.1 terminates when started with both the balanced s.m.p.-candidates, nearly-s.m.p.s and extra-vertices if and only if it terminates when started solely with the balanced s.m.p.-candidates.
It was assumed (personal communication), at least for dimension $s = 1$, that the balancing factors for transition matrices occurring in subdivision theory[2] are always equal to 1. While it is not hard to find counterexamples in dimensions $s > 1$, the claim is also not valid in the univariate case, as Example 4.3 shows. Readers unfamiliar with subdivision schemes may skip Example 4.3.

Example 4.3. Let S be the univariate subdivision scheme defined by the mask a and the dilation matrix M given by

$$a = \frac{1}{12}\begin{bmatrix} 3 & 3 & 4 & 3 & 3 & 4 & 3 & 3 & 4 & 3 & 3 \end{bmatrix}^T, \quad M = -3.$$

The basic limit function can be seen in Figure 4. Taking the digit set $D = \{-2, -1, 0\} = M[0,1) \cap \mathbb{Z}$, we construct the set $\Omega_C = \{-4, -3, -2, -1, 0, 1\}$ (using [12, Lemma 3.8]) and the corresponding transition matrices $T_d = \left[a(\alpha - M\beta)\right]_{\alpha,\beta \in \Omega_C}$, $d \in D$. The restriction of the transition matrices to the space V of first order differences with basis

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

[2]Subdivision schemes are computational means for generating finer and finer meshes in $\mathbb{R}^s$, usually in dimension $s = 1, 2, 3$. At each step of the subdivision recursion, the topology of the finer mesh is inherited from the coarser mesh and the coordinates $c^{(n+1)}$ of the finer vertices are computed by local averages of the coarser ones $c^{(n)}$ by $c^{(n+1)} = Sc^{(n)} = \sum_{\alpha \in \mathbb{Z}^s} a(\cdot - M\alpha)c^{(n)}(\alpha)$. See [12] for a more thorough explanation.

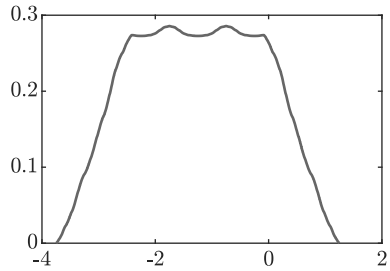

Fig. 4. The basic limit function for the subdivision scheme from Example 4.3.

yields the set of matrices $\mathcal{T}|_V = \{T_{-2}|_V,\ T_{-1}|_V,\ T_0|_V\}$ with

$$T_{-2}|_V = \frac{-1}{12}\begin{bmatrix} 0 & 0 & 0 & 3 & 0 \\ 3 & 0 & 1 & 2 & 0 \\ 2 & 0 & 2 & 1 & 0 \\ 1 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},\ T_{-1}|_V = \frac{-1}{12}\begin{bmatrix} 0 & 0 & 0 & 0 & 3 \\ 0 & 3 & 0 & 1 & 2 \\ 1 & 2 & 0 & 2 & 1 \\ 2 & 1 & 0 & 3 & 0 \\ 3 & 0 & 0 & 0 & 0 \end{bmatrix},\ T_0|_V = \frac{-1}{12}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 1 \\ 0 & 1 & 2 & 0 & 2 \\ 0 & 2 & 1 & 0 & 3 \\ 0 & 3 & 0 & 0 & 0 \end{bmatrix}.$$

For the s.m.p.s $\Pi_1 = T_{-2}T_{-1}T_{-1}|_V$ and $\Pi_2 = T_{-1}T_{-1}T_0|_V$ with balancing vector $[\,1 \quad 9/10\,]$, the original invariant polytope algorithm terminates after 4 iterations. Without balancing the original invariant polytope algorithm does not terminate.

Example 4.4 shows the advantage of the new balancing procedure in connection with nearly-s.m.p.s.

Example 4.4. Given $E_1 = \begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix}$, $E_2 = \begin{bmatrix} 2 & 0 \\ 2 & 1 \end{bmatrix}$, the irreducible set $\mathcal{E} = \{E_1, E_2\}$ has E_2E_1 as an s.m.p. and $\rho(\mathcal{E}) = 2.5396\ldots$. Assuming we start the modified invariant polytope algorithm 4.1 with that candidate and the nearly-s.m.p. E_2, with corresponding leading eigenvectors $v_1^{(0)} = [\,0.9121\ldots \quad 0.4100\ldots\,]^T$, $v_2^{(0)} = [\,0.4472\ldots \quad 0.8944\ldots\,]^T$ and leading dual eigenvectors $v_1^* = [\,0.9958\ldots \quad 0.2238\ldots\,]^T$, $v_2^* = [\,2.2361\ldots \quad 0.0000\ldots\,]^T$. For the balancing procedure as described in [19, Remark 3.7] we need to find numbers $\alpha_1, \alpha_2 > 0$ such that for some $h \in \mathbb{N}$, say $h = 10$, $q_{1,2} = \sup_{z\in\tilde{\mathcal{E}}^h\{v_1^{(0)}, v_1^{(1)}\}} |(v_2^*, z)| = 2.0395\ldots$ and $q_{2,1} = \sup_{z\in\tilde{\mathcal{E}}^h\{v_2^{(0)}\}} |(v_1^*, z)| = 0.8196\ldots$ the following two inequalities hold

$$\alpha_1 \cdot 2.0395\ldots = \alpha_1 q_{1,2} < \alpha_2$$
$$\alpha_2 \cdot 0.8196\ldots = \alpha_2 q_{2,1} < \alpha_1.$$

This is clearly impossible since $2.0 \times 0.8 > 1$. Because there are no admissible balancing factors for $h = 10$, there are no admissible balancing factors for $h > 10$ [19, Section 3].

Since E_1E_2 is a dominant s.m.p., the modified invariant polytope algorithm 4.1 terminates if it is started only with that candidate, and thus, there exist balancing factors such that the the modified invariant polytope algorithm terminates when started with E_2E_1 and E_1, e.g. $\alpha_1 = 1$, $\alpha_2 \simeq 0.95$ as given by our new method.

4.6 Select new children – Natural selection of vertices (15)

In the original invariant polytope algorithm 1.4, in every iteration all vertices generated in the last iteration, which were not mapped inside the polytope, were used to construct new vertices. In the modified invariant polytope algorithm 4.1 we only take a subset of those. We choose the vertices

under the mild condition that

$$\text{for every } n \in \mathbb{N}, \text{ every vertex of } \{\tilde{A}_j\}^n V_0 \text{ eventually will be selected,} \tag{20}$$

given that it is not absorbed already. In other words, we do not forget any vertex to select. This condition is necessary to proof that the modified invariant polytope algorithm and the original invariant polytope algorithm have the same qualitative behaviour in Theorem 4.2.

Two selection strategies turned out to work well:

(a) Choose those vertices that have the largest (e.g. highest decile) norm $\|V_k^+ \cdot\|_2$, where V_k^+ denotes any pseudo-inverse of V_k. In view of Lemma 1.2 (2), the value $\|V_k^+ v\|_2$ is an approximation of $\|v\|_{\mathrm{co}_* V_k}$ and, thus, we may assume that vertices v with high value $\|V_k^+ v\|_2$ are far outside of the polytope $\mathrm{co}_* V_k$.

(b) Choose those vertices whose parent vertex has largest norm with respect to the norm $\|\cdot\|_{\mathrm{co}_* V}$.

With a good selection of new vertices, the polytope $\mathrm{co}_* V_k$ gets large faster, thus, can absorb new vertices faster, and so the number of vertices of the invariant polytope may be smaller. Strategy (a) reduces the number of vertices of the invariant polytope by roughly 20%, strategy (b) by roughly 10%. Since the intermediate bounds b_k for the JSR decreases very slowly when we use strategy (a) only, we use three times (a) and one time (b) in our implementation.

Algorithm 4.5 (Subroutine Natural selection of vertices (15)*).*

Input V_k, **Output** E_{k+1}

---

if $k \neq 0 \bmod 4$ **then** compute $y_v = \|V_k^+ v\|$ for all $v \in \mathcal{A}V_k \setminus \mathcal{V}_k$

else set $y_v = \|w\|_{E_k}$ for all $v \in \mathcal{A}w \setminus \mathcal{V}_k,\ w \in \mathcal{V}_k$

sort E_k

E_{k+1} := Choose 10%, but at least $4 \cdot$ #thread, of the highest values in $E_k$and such that (20) holds

In Algorithm 15, we denote with #thread the number of available threads of the computer. The natural selection of new vertices also makes the modified invariant polytope algorithm 4.1 applicable for problems with a large number of matrices, since it ensures that the number of norms to be computed in each iteration is reasonably small.

4.7 Simplified polytope (16)

In each iteration k we take a subset $W_k \subseteq V_k$ of vertices which are used to compute the norms in step (17) for the vertices in E_{k+1} due to 2 reasons.

Firstly, in some examples the vertices constructed by the modified invariant polytope algorithm 4.1 are very near to each other, i.e. are at distances in the order of the machine epsilon. Those vertices are irrelevant for the size of the polytope and so we disregard them. This also protects against stability problems in the LP-programming part, since for simplices with vertices very near to each other, LP-solvers perform very badly. This phenomenon happens frequently when there are multiple s.m.p.s.. In our implementation we use a variable threshold in (16) when determining which vertices of the polytope we use in the computation of the norm.

Secondly, as we will see in the proof of Theorem 4.2, in order to obtain intermediate bounds b_{k+1} for the JSR, we are only allowed to choose vertices whose children are selected for its norm to be computed, or whose children norms are already computed, i.e. it must be satisfied that

$$\tilde{\mathcal{A}} W_k \subseteq V_k \cup E_{k+1}. \tag{21}$$

It would also be possible to choose a polytope $W(v)$ for each norm $\|v\|_{\mathrm{co}_* W(v)}$ we need to compute, since for each $v \in \mathbb{R}^s$ we only need $s+1$ vectors from V to compute the norm $\|v\|_{\mathrm{co}_* V}$

exactly. Unfortunately we have no idea so far, how to select a good subset of V_k in a reasonable amount of time, i.e. faster than the computation of the norm would take.

4.8 Parallelisation (17) & (18)

This is one of the main differences to the original implementation – the idea is already developed in [21, Algorithm 5.1]. Instead of testing each vertex one after another, and adding it immediately to the set of vertices V_k if it is outside of the polytope, we compute the norms of all selected vertices from step (15) with respect to the same polytope. Afterwards we add all vertices which are outside of the polytope at once to the set V_k.

This clearly leads to larger polytopes, in our examples the number of vertices increases by 10%, but this is compensated by the fact that we can parallelise the computations of the norms. The speed-up is nearly linear in the number of available threads. Since the linear programming model does not change, we can speed up this part further by warm starting the linear programming problems, i.e. we reuse the solutions obtained from the computations of the other vertices. If there are no suitable candidates to warm start with, we still can speed up the LP-problem by starting the search for the solution at the nearest vertex point of the polytope W. The speed-up from warm starting is roughly 50-70%.

4.9 Norm classification (17)

Before computing the exact norm of a vector $A_j v$, we try to determine the relative position (inside or outside of the polytope) using the estimates in Lemma 1.2. If a vertex is proven to be inside or outside of the polytope, we do not have to compute its exact norm anymore. Unfortunately, these estimates are quite rough and fail to determine the position for most vertices, except in case (P) where Lemma 1.2 (5) gives very good estimates.

Algorithm 4.6 (Subroutine Norm classification and adding of vertices (17) (18)*).*

Input E_{k+1}, **Output** V_{k+1}

$V_{k+1} := V_k$
for $v \in E_{k+1}$
  Classify $\|v\|_{\mathrm{co}_* W_k}$ using Lemma 1.2
  If v is outside of $\mathrm{co}_* V_k$ **then** $N(v) := \infty$
  else if v is inside of $\mathrm{co}_* V_k$ **then** $N(v) := 0$
  else $N(v) := \|v\|_{\mathrm{co}_* W_k}$
  If $N(v) > 1 - \epsilon$ **then** $V_{k+1} = V_{k+1} \cup v$

4.10 Spectral radius based stopping criterion (19)

The spectral radius based stopping criterion is used to find better s.m.p.-candidates, in case the chosen s.m.p.-candidates Π_r are no s.m.p.s. If the s.m.p.-candidates Π_r are s.m.p.s, then all intermediately occurring matrix products will have spectral radius less than 1. Unfortunately, the converse is not true.

Algorithm 4.7 (Subroutine Spectral radius based stopping criterion (19)*).*

Input $v \in V_{k+1}$, **Output** Maybe a better s.m.p.-candidate.

---

for $v = \tilde{A}_{j_n} \cdots \tilde{A}_{j_0} v_s^{(0)} \in V_{k+1}$

Compute $\rho = \rho(\tilde{A}_{j_n} \cdots \tilde{A}_{j_0})^{1/n}$

if $\rho > 1$ **then** restart algorithm with s.m.p.-candidate $A_{j_n} \cdots A_{j_0}$

As noted, if the candidates are not s.m.p.s, it can happen that all intermediately occurring matrix products have spectral radius less than 1 and that the modified invariant polytope algorithm 4.1 never stops, see Example 4.8. Nevertheless, this never happened in any non-artificial example. Furthermore, products with larger normalized spectral radius always occurred very fast. Thus, from a practical point of view, the spectral radius based stopping criterion is a better way to check whether the candidates are s.m.p.s than the eigenplane based method described in [17, Proposition 2]. On the other hand, whenever the eigenplane based method [17, Proposition 2] is applicable, it is fail-proof and eventually will strike if an s.m.p.-candidate is not an s.m.p.. Thus, in our implementation of the modified invariant polytope algorithm both stopping criteria are used.

We now illustrate how the new stopping criterion (19) may fail. For that purpose, we introduce for given $\eta \geq 0$ the set

$$\mathcal{M}_\eta = \{(j_n)_n \in \{1, \dots, J\}^{\mathbb{N}} \ : \ \rho(\tilde{A}_{j_m} \cdots \tilde{A}_{j_1})^{1/m} \leq \eta, \quad \forall m \in \mathbb{N}\}.$$

For $\eta = 1$, the products $\tilde{A}_{j_n} \cdots \tilde{A}_{j_1}$, $(j_n)_n \in \mathcal{M}_1$, are exactly the products occurring in the modified invariant polytope algorithm 4.1 until the spectral radius based stopping criterion (19) strikes. The hope would be, that the norms of the products in that sequence stay bounded, i.e. $\exists C > 0$ such that $\|\tilde{A}_{j_n} \cdots \tilde{A}_{j_1}\| < C$ for all $n \in \mathbb{N}$.

Example 4.8. Let $A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$, $B = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$. Clearly $\mathrm{JSR}(\{A, B\}) = \rho(AB)^{1/2} = (\sqrt{5}+1)/2$ and $\{A, B\}$ is irreducible. We choose $\Pi_1 = A$ and $\Pi_2 = B$ as our two (wrong) s.m.p.-candidates, thus, $\rho_c = \rho_r = 1$, $\tilde{\Pi}_1 = \tilde{A} = A$, $\tilde{\Pi}_2 = \tilde{B} = B$, $V_0 = \mathcal{H} = \{v_1^{(0)}, v_2^{(0)}\}$ with $v_1^{(0)} = [\,1 \quad 0\,]^T$, $v_2^{(0)} = [\,0 \quad 1\,]^T$, and since there are no extra vertices, $V_0 = \mathcal{H}$.

Now, we use a (bad) selection procedure of new vertices E_{k+1} in (15) of the natural selection of vertices; namely, we choose only the vertices $A^n v_2$ and $B^n v_1$, $n \in \mathbb{N}$.[3]

We now show that the algorithm constructs an infinitely big polytope, solely with vertices generated by matrix products whose averaged spectral radius is equal to ρ_c. Indeed, for $n \in \mathbb{N}$, applying the sequence of products A^n to the starting vector v_2 we get the sequences of vector $A^n v_2 = [\,n \quad 1\,]^T$, where $\rho(A^n)^{1/n} = 1$. The same calculation shows that $B^n v_1 = [\,1 \quad n\,]^T$ and $\rho(B^n)^{1/n} = 1$. Finally, $\mathrm{co}_- V_k = \mathrm{co}_-\{[\,1 \quad k\,]^T, [\,k \quad 1\,]^T\}$, $k \in \mathbb{N}$.

4.11 Proof for Theorem 4.2

Proof. (i) Let $\delta = 1$. Assume that the original invariant polytope algorithm 1.4 terminates at depth $N \in \mathbb{N}$ with vertices V_N^{orig}, i.e. $\tilde{\mathcal{A}}\,\mathrm{co}_* V_N^{orig} \subseteq \mathrm{co}_* V_N^{orig}$. By construction of the original invariant polytope algorithm 1.4 and by (20), there exists $K \in \mathbb{N}$, $K \geq N$, such that $\mathrm{co}_* V_N^{orig} = \mathrm{co}_* \bigcup_{n=0}^N \tilde{\mathcal{A}}^n V_0 \subseteq \mathrm{co}_* V_K^{mod}$. We claim that $\mathrm{co}_* V_K^{mod}$ is an invariant polytope. By construction of the modified invariant polytope algorithm 4.1, $\mathrm{co}_* V_K^{mod} \subseteq \mathrm{co}_* \bigcup_{k=0}^K \tilde{\mathcal{A}}^k V_0$. By the invariance property of the polytope $\mathrm{co}_* V_N^{orig}$ and by $K > N$, $\mathrm{co}_* \bigcup_{k=0}^K \tilde{\mathcal{A}}^k V_0 = \mathrm{co}_* V_N^{orig}$. It follows that $\mathrm{co}_* V_N^{orig} = \mathrm{co}_* V_K^{mod}$, and thus $\mathrm{co}_* V_K^{mod}$ is an invariant polytope.

[3] Actually, this selection of vertices is neither type (a) or (b) from Section 4.6, nor does it fulfil the necessary condition (21).

The other direction follows similarly.

(*ii*) Assume that $\mathrm{JSR}(\mathcal{A}) < \delta^{-1} \cdot \rho_c$, or equivalently, $\mathrm{JSR}(\tilde{\mathcal{A}}) < \gamma < 1$ for some $\gamma > 0$. [3, Theorem I (b)] implies that $\|\tilde{A}_{i_k} \cdots \tilde{A}_{i_1}\| \to 0$ for any product $\tilde{A}_{i_k} \cdots \tilde{A}_{i_1} \in \tilde{\mathcal{A}}^n$ as $k \to \infty$. Thus, the modified invariant polytope algorithm eventually terminates.

(*iii*) Let $k \in \mathbb{N}_0$. Without loss of generality we assume that $1 < b_{k+1} < b_k$. Let $v \in V_{k+1}$. We need to show that $\|\tilde{A}_j v\|_{\mathrm{co}_* V_{k+1}} \leq b_{k+1}$ for all $j \in \{1, \ldots, J\}$. If $\tilde{A}_j v \in V_{k+1}$, then we trivially get $\|\tilde{A}_j v\|_{\mathrm{co}_* V_{k+1}} \leq 1 < b_{k+1}$. Thus, we assume that $\tilde{A}_j v \notin V_{k+1}$. Let $k' \in \mathbb{N}_0$ be the iteration in which $N(\tilde{A}_j v)$ was computed. By (21), $\mathrm{co}_* W_{k'} \subseteq \mathrm{co}_* V_{k+1}$. Therefore, $\|\tilde{A}_j v\|_{\mathrm{co}_* V_{k+1}} \leq \|\tilde{A}_j v\|_{\mathrm{co}_* V_{k'}} \leq \|\tilde{A}_j v\|_{\mathrm{co}_* W'_k} = N(v)(1+\epsilon)^{-1} \leq b_{k+1}$. □

5 APPLICATIONS AND NUMERICAL RESULTS

In this section we illustrate the modified Gripenberg algorithm 3.1 and the modified invariant polytope algorithm 4.1 with numerical examples. For our tests we use matrices from standard applications, as well as random matrices. We also try to repeat tests previously performed in the literature [4, 5, 7, 17, 19, 28].

The parameters for the various algorithms (ours and others) are chosen such that they terminate after a reasonably short time. For the modified invariant polytope algorithm 4.1 the parameters are chosen such that the modified invariant polytope algorithm terminates at all, hopefully in shortest time. We do not report the exact parameters, since we believe they are of no value for the reader. The tests are performed using an Intel Core i5-4670S@3.8GHz, 8GB RAM with the software Matlab R2017a and Gurobi solver v8.0.[4]

For the tests we report • the dimension *dim* of the matrices, • the duration *time* needed for the computation (this value is only to be understood in magnitudes), • the number of matrices J in the test set $\mathcal{A}$, • the number of vertices $\#V$ of the invariant polytope, • spectral maximizing product(s) *s.m.p.*, and • the number *#tests* of test runs.

5.1 Main results

5.1.1 Modified invariant polytope algorithm. To summarize, we can say that the single-threaded modified invariant polytope algorithm 4.1 is roughly three times faster than the original invariant polytope algorithm 1.4. If the dimension of the matrices is sufficiently large, the parallelised modified invariant polytope algorithm 4.1 scales nearly linearly with the number of available threads (for at least up to 16 threads). More precisely,

- for pairs of random matrices the modified invariant polytope algorithm 4.1 reports the exact value of the JSR in reasonable time up to dimension 25,
- for Daubechies matrices the modified invariant polytope algorithm reports the exact value of the JSR in reasonable time up to dimension 42,
- for non-negative matrices it strongly depends on the problem. For random, sparse, non-negative matrices the modified invariant polytope algorithm works up to dimension 3000 or higher. For the (sparse) matrices arising in the context of code capacities (Section 5.4) the modified invariant polytope algorithm works well only up to dimension 16. On the one hand this is due to the large number of matrices to be considered for these examples, on the other hand the structure of the individual matrices seems to play a role.

5.1.2 Modified Gripenberg algorithm. For the modified Gripenberg algorithm 3.1 we can say, that it finds in almost all cases an s.m.p.. Thus, for fast estimates of the JSR, the modified Gripenberg

[4] (*a*) Our implementation also uses software containing functions from the JSR-Toolbox v1.2b [25]. Permission to use has been kindly granted. (*b*) The Gurobi solver is free for academic use.

algorithm 3.1 may be used independently, e.g. in applications where the parameters where a matrix family has highest/lowest JSR need to be determined. In a second step one then may compute the exact JSR for the found parameters using the modified invariant polytope algorithm.

Clearly, since the computation of the JSR is NP-hard, there must be sets of matrices for which the modified Gripenberg algorithm 3.1 fails[5] and we report mostly these cases together with a comparison with other algorithms. These are

• the *random Gripenberg* algorithm 3.1 described in Remark 3.4, • the *Gripenberg* algorithm, • the *modified invariant polytope* algorithm 4.1 and • the Monte-Carlo type *genetic* algorithm [4].

At least in our test runs, the modified Gripenberg algorithm 3.1 performs best, in the sense that in most cases it returns a correct s.m.p. in fastest time. More precisely, for long s.m.p.s the modified Gripenberg algorithm 3.1 performs best and for large sets of matrices the genetic algorithm and the modified invariant polytope algorithm 4.1 performs best.

5.2 Randomly generated matrices

We first present the behaviour of the modified invariant polytope algorithm 4.1 for pairs of matrices of dimensions 2 to 20 with normally distributed values whose (*a*) matrices have the same 2-norm, (*b*) matrices have the same spectral radius, and (*c*) matrices have the same spectral radius and $\delta = 0.99$ (where δ was the parameter controlling the accuracy of the modified invariant polytope algorithm 4.1, see Section 4.3 (12)). We see in Table 1 that the modified invariant polytope algorithm is applicable for pairs of random matrices up to dimension 25, for which it takes roughly one weekend to complete. For $\delta = 0.95$ the modified invariant polytope algorithm is comparable to Gripenberg's algorithm.

Although the modified invariant polytope algorithm 4.1 produces polytopes with roughly twice as much vertices compared to the same test with the original invariant polytope algorithm in [17, Table 2], it still works very well for matrices of dimension 20.

Table 1. Computation of the JSR for random pairs of matrices using the modified invariant polytope algorithm 4.1. δ: accuracy parameter for the modified invariant polytope algorithm (12), *dim*: dimension of the matrices, *#V*: number of vertices of the invariant polytope, *time*: time needed to compute the invariant polytope, *J*: number of matrices, *#test*: number of test runs.
[†]We print median values, since there are always outliers if $\delta = 1$. The average values are roughly 100 times bigger.

$J = 2$, *#test* $= 20$, *median values*[†]						
	(*a*) $\delta = 1$ *equal norm*		(*b*) $\delta = 1$ *equal spectral radius*		(*c*) $\delta = 0.99$ *equal spectral radius*	
dim	*time*	*#V*	*time*	*#V*	*time*	*#V*
2	1.1 *s*	5·2	1.2 *s*	6·2	0.2 *s*	5·2
4	1.4 *s*	17·2	1.8 *s*	77·2	0.8 *s*	19·2
6	2.0 *s*	47·2	2.5 *s*	130·2	1.5 *s*	47·2
8	2.5 *s*	100·2	3.9 *s*	220·2	2.1 *s*	98·2
10	4.9 *s*	270·2	5.1 *s*	320·2	3.3 *s*	220·2
12	4.7 *s*	280·2	11 *s*	770·2	6.6 *s*	570·2
14	8.4 *s*	510·2	21 *s*	1100·2	12 *s*	800·2
16	25 *s*	1100·2	33 *s*	1400·2	25 *s*	1000·2
18	90 *s*	2100·2	200 *s*	2500·2	44 *s*	1600·2
20	295 *s*	3100·2	5000 *s*	6200·2	800 *s*	3900·2

[5]since the modified Gripenberg algorithm has polynomial complexity,

Random matrices with non-negative entries are a worthy test case, since the computation of the invariant polytope (i.e. the main loop in the modified invariant polytope algorithm 4.1) always finishes after a few seconds, nearly regardless of the dimension. Since the implementation is not optimized for such high dimensions, the modified invariant polytope algorithm still needs some minutes to terminate, mostly due to the preprocessing steps (11)-(14). For sparse matrices with non-negative entries, the modified invariant polytope algorithm 4.1 performs slightly worse, but is still applicable up to dimension 2000 or higher. Again, it is very likely that it still works for even larger matrices if the implementation were optimized for such matrices, see Table 2 for the results. We again give the median values. The average values for these cases are roughly 10% higher. Another benchmark for non-negative matrices is presented in Section 5.4.

Table 2. Computation of the JSR using the modified invariant polytope algorithm 4.1 for random pairs of matrices with non-negative entries. *dim*: dimension of the matrices, J: number of matrices, $\#test$: number of test runs. *time*: time needed to compute the invariant polytope, $\#V$: number of vertices of the polytope. †We print the median values, since there are always outliers if $\delta = 1$. The average values are roughly 100 times bigger. ††Since the matrices are random, most of the sparse matrices have non-trivial invariant subspaces which reduces the effective dimension of the matrices by roughly 10%. †††Most cones have 8 or 16 vertices, because the algorithm terminates after 3 or 4 iterations. The algorithm does not check whether all of these vertices are really outside of the polytope.

$J = 2$, $\#test = 20$, *non-negative entries, equal spectral radius, median values*†								
	0% sparsity		*90% sparsity*		*98% sparsity*		*99% sparsity*	
dim††	*time*	$\#V$†††	*time*	$\#V$†††	*time*	$\#V$	*time*	$\#V$
20	0.3 *s*	7	1.7 *s*	42				
50	0.3 *s*	8	1.6 *s*	50	2.2 *s*	50		
100	0.4 *s*	8	0.8 *s*	25	17 *s*	1300		
200	0.5 *s*	8	1.0 *s*	23	5.0 *s*	220	110 *s*	2600
500	1.2 *s*	8	1.8 *s*	16	7.7 *s*	90	26 *s*	310
1000	6.3 *s*	8	11 *s*	16	30 *s*	45	72 *s*	110
2000	35 *s*	8	72 *s*	16	35 *s*	8	290 *s*	64

In Table 3 we see how the modified Gripenberg algorithm 3.1 performs on random matrices with equally distributed values in $[-5,\ 5]$ to mimic the test in [4, Section 4.2]. Interestingly, the genetic algorithm performs very bad, as does the *random* modified Gripenberg algorithm. We report the *succes*-rate, i.e. how often the algorithms did find an s.m.p. in percent.

5.3 Handpicked generic matrices

Example 5.1. Let

$$X_1 = \begin{bmatrix} \frac{15}{92} & \frac{-73}{79} \\ \frac{56}{59} & \frac{89}{118} \end{bmatrix}, \; X_2 = \begin{bmatrix} \frac{-231}{241} & \frac{-143}{219} \\ \frac{103}{153} & \frac{-38}{65} \end{bmatrix}.$$

The set $\mathcal{X} = \{X_1, X_2\}$ has an s.m.p. of length 119 with normalized spectral radius $\mathrm{JSR}(\mathcal{X}) \simeq 1.01179$. Gripenberg's algorithm finds an s.m.p. after an evaluation of ~630k products, taking roughly ten minutes. Both the modified Gripenberg algorithm, as well as the genetic algorithm fail. The modified invariant polytope algorithm 4.1 finds an s.m.p. after less than one minute. The test results are in Table 4.

Example 5.2 is of interest because it is a rather simple family of two matrices with an arbitrary long s.m.p..

Table 3. Performance of various algorithms searching for s.m.p.s. We use the modified invariant polytope algorithm to test, whether the found s.m.p.-candidates are indeed s.m.p.s. *dim*: dimension of the matrices, *J*: number of matrices, *success*: percentage of how often a correct s.m.p. is found. *#test*: number of test runs. *time*: time needed by the algorithm.

$\#tests = 100$						
	$J = 2, dim = 2$		$J = 4, dim = 4$		$J = 8, dim = 8$	
Algorithm	*success*	*time*	*success*	*time*	*success*	*time*
mod. invariant polytope	100%	1.1 *s*	100%	4.3 *s*	100%	40.0 *s*
mod. Gripenberg	100%	1.9 *s*	100%	4.1 *s*	100%	5.4 *s*
random Gripenberg	100%	1.8 *s*	99%	3.8 *s*	82%	4.3 *s*
Gripenberg	100%	3.8 *s*	100%	20.3 *s*	100%	82.1 *s*
brute force	100%	180 *s*	98%	180.0 *s*	74%	180.0 *s*
genetic	100%	7.1 *s*	97%	9.3 *s*	87%	12.0 *s*

Table 4. Performance of various algorithms searching for s.m.p.s for a particular hard problem. For the test set $\mathcal{X}$ (Example 5.1) all fast algorithms fail. *dim*: dimension of the matrices, *lower bd.*: computed lower bound for the JSR, *J*: number of matrices, *time*: time needed by the algorithm.

Testset	*Algorithm*	*lower bd.*	*time*
$\mathcal{X}$	mod. invariant polytope	1.01179 . . .	40 *s*
$J = 2$	mod. Gripenberg	1.01130 . . .	4 *s*
$dim = 2$	random Gripenberg	1.01172 . . .	10 *s*
	Gripenberg	1.01179 . . .	580 *s*
	genetic	1.01130 . . .	8 *s*

Example 5.2. Let $n \in \mathbb{N}$, $C_0 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ and $C_n = \begin{bmatrix} 0 & 0 \\ \frac{1}{n}e^{1+\frac{1}{n}} & 0 \end{bmatrix}$, Then $C_0^n C_n$ is an s.m.p. for the set $\mathcal{C}_n = \{C_0, C_n\}$ with $\mathrm{JSR}(\mathcal{C}_n) = e^{1/n}$.

The genetic algorithm fails for most matrices of that family. All other algorithms report the correct s.m.p. in less than 5 *s*. The test results are in Table 5.

Proof for Example 5.2. Define $\tilde{C}_n = \begin{bmatrix} 0 & 0 \\ n & 0 \end{bmatrix}$, $n \in \mathbb{N}$. A product of C_0 and $\tilde{C}_n$ is non-zero if and only if it is of the form $C_0^{i_1}\tilde{C}_n C_0^{i_2}\tilde{C}_n \cdots \tilde{C}_n C_0^{i_m}$. Since the spectral radius does not change under cyclic permutation, we can assume that the product is of the form $C_0^{i_1}\tilde{C}_n C_0^{i_2}\tilde{C}_n \cdots C_0^{i_m}\tilde{C}_n$. A (lengthy) straightforward computation shows that the normalized spectral radius of this product is $(n^m \prod_{j=1}^m i_j)^{1/(m+\sum_{j=1}^m i_j)}$. Taking the gradient with respect to i and setting it to zero, we immediately get that all i_j must be equal. Thus, the normalized spectral radius of all finite products is maximized with a product of the form $C_0^m \tilde{C}_n$ whose normalized spectral radius equals $mn^{1/(1+m)}$. For fixed $m \in \mathbb{N}$ this term has its maximum at $n = \frac{1}{m}e^{1+1/m}$. Thus, $C_0^n C_n$ is the product with largest normalized spectral radius under all finite products. Using (2) we conclude that $\mathrm{JSR}(\mathcal{C}) = \rho(C_0^n C_n)^{1/(n+1)} = (e^{(n+1)/n})^{1/(n+1)} = e^{1/n}$. □

5.4 Capacity of codes with forbidden difference sets

In some electromagnetic recording systems, the bit error rate is often dominated by a small set of certain *forbidden difference patterns D*. Thus, one needs to construct sets of allowed words with

Table 5. Performance of various algorithms searching for s.m.p.s. For the test sets C_n (Example 5.2) the genetic algorithm mostly fails. *dim*: dimension of the matrices, *lower bd.*: computed lower bound for the JSR, *J*: number of matrices, *s.m.p.*: an s.m.p., *time*: time needed to compute the invariant polytope,

Test set	*Algorithm*	*lower bd.*	*time*
C_{15}	mod. invariant polytope	1.0689 . . .	1.7 *s*
$J = 2$	mod. Gripenberg	1.0689 . . .	3.3 *s*
$dim = 2$	random Gripenberg	1.0689 . . .	3.2 *s*
$s.m.p. = C_0^{15}C_{15}$	Gripenberg	1.0689 . . .	0.1 *s*
	genetic	1.0689 . . .	7.0 *s*
C_{30}	mod. invariant polytope	1.0338 . . .	2.5 *s*
$J = 2$	mod. Gripenberg	1.0338 . . .	4.0 *s*
$dim = 2$	random Gripenberg	1.0338 . . .	4.3 *s*
$s.m.p. = C_0^{30}C_{30}$	Gripenberg	1.0338 . . .	0.1 *s*
	genetic	1.0$\underline{=}$215 . . .	6.6 *s*
C_{60}	mod. invariant polytope	1.0168 . . .	4.0 *s*
$J = 2$	mod. Gripenberg	1.0168 . . .	3.1 *s*
$dim = 2$	random Gripenberg	1.0168 . . .	4.3 *s*
$s.m.p. = C_0^{60}C_{60}$	Gripenberg	1.0168 . . .	0.1 *s*
	genetic	1.0$\underline{0}$00 . . .	6.3 *s*

values in $\{0, 1\}$, all of whose possible differences do not yield such a forbidden pattern. Clearly, one wants codes which constrain the number of all possible patterns as least as possible. We are interested in how constraining a given forbidden difference pattern is, which we denote as the *capacity* $\operatorname{cap} D \in [0,\ 1]$. The larger the capacity, the better. This problem can be expressed in terms of the JSR of a finite set of matrices. See [28] for more details. The occurring matrices in this application only have entries in $\{0, 1\}$, but their dimension, as well as the number of matrices increases exponentially with the length of the forbidden difference patterns, e.g. for $D = \{\circ + - \}$ the capacity of D is given by

$$\operatorname{cap} D = \log_2 \operatorname{JSR}\left(\left\{\begin{bmatrix}1&0&1&0\\0&0&0&0\\0&1&0&1\\0&1&0&1\end{bmatrix}, \begin{bmatrix}1&0&1&0\\0&0&1&0\\0&1&0&0\\0&1&0&1\end{bmatrix}, \begin{bmatrix}1&0&1&0\\1&0&0&0\\0&0&0&1\\0&1&0&1\end{bmatrix}, \begin{bmatrix}1&0&1&0\\1&0&1&0\\0&0&0&0\\0&1&0&1\end{bmatrix}\right\}\right).$$

We use the modified invariant polytope algorithm to compute the capacities for the forbidden difference patterns D taken from [28, p. 10], [4, Table 1], [7, p. 6] and for difference sets with the additional symbol ± , denoting +1 and −1, discussed in [7, Section v]. Nearly all of these capacities were not known exactly before.

For most difference sets D, there are several s.m.p.s., that not only share the same leading eigenvalue but also the same eigenvector. Due to this reason, the modified invariant polytope algorithm 4.1 sometimes only gives a bound for the JSR up to the accuracy in which we can compute the norms $\|\tilde{A}_j v\|_{\mathrm{co}_* W}$. We implemented the Matlab routine `codecapacity` which computes the set of matrices needed for the JSR computation for a given difference set D. It works for reasonably small difference sets, and theoretically also for difference words with entries in $\{-K, \ldots, K\}, K \in \mathbb{N}$.

The exact computation of the capacity using the modified invariant polytope algorithm 4.1 was only possible if we used the estimates for the Minkowski norm in Lemma 5 (1.2), which reduced the norms to be computed by a factor of 100.

Table 6. Capacity of various difference sets D. $\epsilon = 10^{-10}$: computational accuracy, cap(D): capacity, D: set of forbidden differences, *dim*: dimension of the matrices, J: number of matrices, $\#V$: number of vertices of the invariant polytope, *s.m.p.*: an s.m.p..
[?]For some sets D the modified invariant polytope algorithm 4.1 did not terminate, thus the given product is not proven to be an s.m.p.

D	*s.m.p.*	cap(D)	$\#V$	J	*dim*
$\pm\,\pm$	B_2B_3	$1/2$	$3\cdot 2$	4	2
$\circ\,\pm$	B_3	0	$2\cdot 2$	4	2
$\circ + -$	B_4B_1	$0.6942\ldots$	$45\cdot 2$	4	4
$\circ + +$	B_2[?]	$0.6942\ldots + [0,\epsilon]$	$25\cdot 2$	4	4
$\circ\,\pm\,\pm$	B_1B_2	$1/2$	$37\cdot 2$	16	4
$\pm\,\pm\,\pm$	$B_6B_4B_1$	$2/3$	$19\cdot 2$	16	4
$+ - + -$	B_1B_2	$0.9468\ldots$	$86\cdot 2$	2	8
$+ + + -$	B_1B_2	$0.9005\ldots$	$40\cdot 2$	2	8
$+ + + +$	B_1	$0.9468\ldots$	$84\cdot 2$	2	8
$\circ + - +$	B_3	$0.8791\ldots$	$43\cdot 2$	4	8
$\circ + + -$	B_3	$0.8113\ldots$	$46\cdot 2$	4	8
$\circ + + +$	B_1	$0.8791\ldots$	$46\cdot 2$	4	8
$\circ + +\,\pm$	$B_1^2B_2^2$	$0.7396\ldots$	$244\cdot 2$	16	8
$\circ + \circ +$	$B_4B_{11}^2B_{13}B_6^2$	$0.7298\ldots$	$804\cdot 2$	16	8
$\circ + \circ\,\pm$	$B_{16}B_{52}B_{103}$[?]	$2/3 + [0,\epsilon]$	$23152\cdot 2$	256	8
$\pm\,\pm\,\pm\,\pm$	$B_{86}B_{52}B_{16}B_1$	$3/4$	$357\cdot 2$	256	8
$\circ + - + \circ$	$B_{11}B_{13}$	$0.9163\ldots$	$1721\cdot 2$	16	16
$\circ + + + \circ$	B_4B_6	$0.9163\ldots$	$4559\cdot 2$	16	16
$\circ + + + + \circ$	B_2[?]	$0.9614\ldots + [0,\epsilon]$	$17902\cdot 2$	16	32
$+ + + + + - \circ$	B_3	$0.9761\ldots$	$992\cdot 2$	4	64

Table 7. Performance of various algorithms searching for s.m.p.s for a particular hard problem. The modified Gripenberg algorithm 3.1 fails for the set of matrices corresponding to the forbidden difference set D_4 in Section 5.4. *dim*: dimension of the matrices, *lower bd.*: computed lower bound for the JSR, J: number of matrices, *time*: time needed by the algorithm.

Testset	*Algorithm*	*lower bd.*	*time*
$D_4 = \{\circ\circ + \circ -\}$	mod. invariant polytope	$1.6736\ldots$	40 *s*
$J = 256$	mod. Gripenberg	$1.6\underline{6}63\ldots$	2 *s*
dim = 16	random Gripenberg	$1.6\underline{6}63\ldots$	2 *s*
	Gripenberg	$1.6\underline{6}63\ldots$	60 *s*
	genetic	$1.6736\ldots$	10 *s*

The difference set $D_4 = \{\circ\circ + \circ -\}$, taken from [4, Table 1], is a good test case for the modified Gripenberg algorithm, since the computation of the capacity translates to the JSR of a set with 256 matrices of dimension 16. As one can expect, Gripenberg's algorithm fails to find an s.m.p., also the modified Gripenberg algorithm 3.1 fails. The genetic algorithm in most cases finds a better product than the one found by Gripenberg's algorithm. The modified invariant polytope algorithm 4.1 also finds that better product after a while, but it did not terminate in reasonable time. Thus, the exact capacity, and whether an s.m.p. exists is still unknown. The test results are in Table 6 and 7.

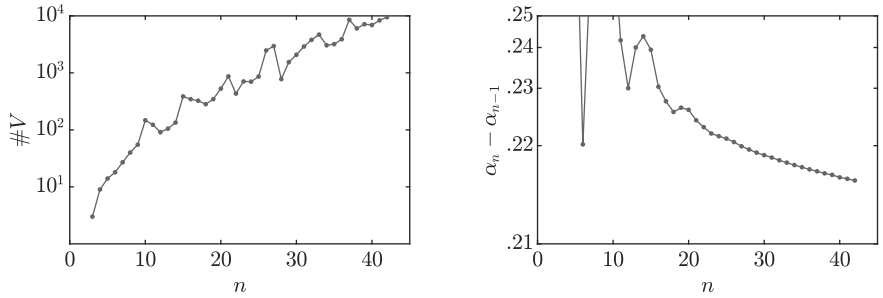

Fig. 5. Left: Number of vertices of the polytope $\#V$ against index of Daubechies wavelet D_n. Right: Difference of regularities α of consecutive Daubechies wavelets.

5.5 Hölder exponents of Daubechies wavelets

An important application of the JSR is the computation of the regularity of *refinable functions*. These are functions $\phi \in C_0(\mathbb{R}^s)$ which fulfil a functional equation of the form $\phi(x) = \sum_{\alpha \in \mathbb{Z}^s} a(\alpha)\phi(2x - \alpha)$, $x \in \mathbb{R}$, with $a \in \ell_0(\mathbb{Z}^s)$. We use the modified invariant polytope algorithm 4.1 to compute the Hölder regularity of the Daubechies wavelets D_n [14]. The regularity of D_2, D_3, and D_4 was computed by Daubechies and Lagarias [15], Gripenberg [16] computed it for $D_5, \ldots, D_8$, then Guglielmi and Protasov [19], as a demonstration of the original invariant polytope algorithm, computed the regularity of $D_9, \ldots, D_{20}$. Now with the modified invariant polytope algorithm, we can compute the Hölder regularity for Daubechies wavelets up to D_{42}.

As noted in [19, Section 6.2], the polytopes generated by these matrices are very flat and the introduction of nearly-candidates and extra-vertices tremendously increases the performance of the invariant polytope algorithm. Respectively, using the wrong set of nearly-candidates, the modified invariant polytope algorithm did not terminate at all. These cases are marked with † in Table 8. The right nearly-candidates and extra-vertices, i.e. good values for τ, T, B_{nearly} and B_{extra}, were merely found by trial and error. We report the number of extra-vertices and the vertices of the roots from the nearly-s.m.p.s together under *#Extra-V*. The number of the invariant polytopes vertice's is depicted in Figure 5 (left side).

Remark 5.3. With the new values for D_{21} to D_{42} we can refine the observation in [18], that the differences of Hölder regularities $\alpha_n - \alpha_{n-1}$ seem to converge towards a value of 0.21 or maybe even 0.2, see Figure 5 (right side).

6 CONCLUSION AND FURTHER WORK

6.1 Conclusion

The modified Gripenberg algorithm 3.1 together with the modified invariant polytope algorithm 4.1 can compute the exact value of the JSR in a short time (less than 30 minutes) for most matrix families up to dimension 22, in some cases even up to dimension 40. For matrices with non-negative entries, the modified invariant polytope algorithm may work up to a dimension of 3000. Even more, since the modified Gripenberg algorithm 3.1 finds in almost all cases a correct s.m.p., it may be used alone for fast estimates of the JSR in time critical applications.

6.2 Further work

From the mathematical point of view, the question why the modified Gripenberg algorithm 3.1 works so well is of interest, in particular why it works mostly better than the *random* Gripenberg algorithm. It also may be useful to search for better estimates for the Minkowski norms, e.g. with

Table 8. Hölder regularity of Daubechies wavelets. α: Hölder regularity of Daubechies wavelet, D_n: index of Daubechies wavelet, *#V*: number of vertices of the invariant polytope, *#Extra-V*: number of extra-vertices including those from nearly-s.m.p.s., *s.m.p.*: an s.m.p., *time*: time needed to compute the invariant polytope. For the cases marked with †, using the wrong set of nearly-candidates, the algorithm did not terminate at all.

D_n	*s.m.p.*	*#Extra-V*	*#V*	*time*	α
2	B_0	0	0·2	$< 5\,s$	0.55001 . . .
3	B_0	0	3·2	$< 5\,s$	1.08783 . . .
4	B_0	2	9·2	$< 5\,s$	1.61793 . . .
5	B_0 and B_1	2	14·2	$< 5\,s$	1.96896 . . .
6	B_0 and B_1	3	18·2	$< 5\,s$	2.18914 . . .
7	B_0 and B_1	4	27·2	$< 5\,s$	2.46041 . . .
8	B_0 and B_1	5	40·2	$< 5\,s$	2.76082 . . .
9	B_0 and B_1	6	55·2	$< 5\,s$	3.07361 . . .
10	$B_0^2B_1^2$	5	147·2	$< 5\,s$	3.36139 . . .
11	B_0 and B_1	8	123·2	$7\,s$	3.60347 . . .
12	B_0 and B_1	9	91·2	$7\,s$	3.83348 . . .
13	B_0 and B_1	10	105·2	$6\,s$	4.07348 . . .
14	B_0 and B_1	11	134·2	$8\,s$	4.31676 . . .
15	$B_0^4B_1^2$	11	386·2	$6\,s$	4.55612 . . .
16	$B_0^2B_1^2$	12	346·2	$7\,s$	4.78644 . . .
17	B_0 and B_1	14	324·2	$5\,s$	5.01380 . . .
18	B_0 and B_1	15	282·2	$8\,s$	5.23917 . . .
19	B_0 and B_1	16	346·2	$9\,s$	5.46532 . . .
20	B_0 and B_1	17	529·2	$12\,s$	5.69108 . . .
21	$B_0^2B_1^2$	17	868·2	$15\,s$	5.91500 . . .
22†	$B_0^2B_1^4$	22	433·2	$9\,s$	6.13779 . . .
23	B_0 and B_1	20	707·2	$18\,s$	6.35958 . . .
24	B_0 and B_1	21	701·2	$16\,s$	6.58096 . . .
25	B_0 and B_1	22	861·2	$20\,s$	6.80198 . . .
26	$B_0^4B_1^2$	22	2471·2	$73\,s$	7.02250 . . .
27	$B_0^2B_1^2$	29	2952·2	$60\,s$	7.24241 . . .
28†	$B_0^2B_1^6$	105	777·2	$24\,s$	7.46187 . . .
29	B_0 and B_1	26	1545·2	$39\,s$	7.68091 . . .
30	B_0 and B_1	27	2078·2	$64\,s$	7.89962 . . .
31	B_0 and B_1	29	2898·2	$190\,s$	8.11801 . . .
32	$B_0^2B_1^2$	29	3791·2	$760\,s$	8.33605 . . .
33†	$B_0^2B_1^2$	30	4692·2	$1330\,s$	8.55379 . . .
34	B_0 and B_1	32	3047·2	$628\,s$	8.77123 . . .
35	B_0 and B_1	33	3191·2	$727\,s$	8.98841 . . .
36	B_0 and B_1	34	3887·2	$881\,s$	9.20533 . . .
37	$B_0^6B_1^2$	70	8529·2	$6503\,s$	9.42202 . . .
38	$B_0^2B_1^2$	38	6035·2	$3540\,s$	9.63847 . . .
39	$B_0^2B_1^4$	40	7142·2	$3900\,s$	9.85474 . . .
40	B_0 and B_1	38	6909·2	$5550\,s$	10.07073 . . .
41	B_0 and B_1	39	8343·2	$8743\,s$	10.28656 . . .
42	B_0 and B_1	40	9508·2	$16373\,s$	10.50220 . . .

orthant-monotonic norms, which would lead to a considerable speed up of the modified invariant polytope algorithm.

From the algorithmic point of view, the modified invariant polytope algorithm could be made faster by using approximate solutions to the LP-problem when computing the Minkowski-norms, since the exact value of the norms is of minor interest — for the modified invariant polytope algorithm it is enough to know whether a point is inside or outside of the polytope.

We plan to implement the case (C) of complex leading eigenvalue in the near future and optimize the modified invariant polytope algorithm for a large number of parallel threads. Case (C) occurs seldom, in the sense that we did not encounter a set of matrices of practical interest with complex leading eigenvectors yet.

ACKNOWLEDGMENTS

The author is grateful for the hospitality, help and encouragement of Prof. V. Yu. Protasov. The work is supported by the Austrian Science Fund under Grant P 28287 (www.fwf.ac.at) and by "Vienna Scientific Cluster" (VSC) for providing computational resource

I would like to thank the referees for several helpful suggestions which greatly improved the presentation of this paper.

REFERENCES

[1] A. Ahmadi, Raphaël Jungers, Pablo A. Parrilo, and Mardavjij Roozbehani. 2011. Joint Spectral Radius and Path-Complete Graph Lyapunov Functions. *SIAM J. Control Optim.* 52, 1 (2011), 687–717.

[2] N. E. Barabanov. 1988. Lyapunov indicator for discrete inclusions I–III. *Autom. Remote Control* 49, 2 (1988), 152–157.

[3] Marc A. Berger and Yang Wang. 1992. Bounded semigroups of matrices. *Linear Alg. Appl.* 166 (1992), 21–27.

[4] Vincent D. Blondel and Chia-Tche Chang. 2011. A genetic algorithm approach for the approximation of the joint spectral radius. https://perso.uclouvain.be/chia-tche.chang/code.php.

[5] Vincent D. Blondel and Chia-Tche Chang. 2013. An experimental study of approximation algorithms for the joint spectral radius. *Numer. Algor.* 64 (2013), 181–202.

[6] Vincent D. Blondel and Raphaël Jungers. 2008. On the finiteness property for rational matrices. *Linear Alg. Appl.* 428, 10 (2008), 2283–2295.

[7] Vincent D. Blondel, Raphaël Jungers, and Vladimir Yu. Protasov. 2006. On the Complexity of Computing the Capacity of Codes That Avoid Forbidden Difference Patterns. *IEEE Trans. Inf. Theory* 52 (2006), 5122–5127.

[8] Vincent D. Blondel, Raphaël Jungers, and Vladimir Yu. Protasov. 2010. Joint spectral characteristics of matrices: a conic programming approach. *SIAM J. Matr. Anal. Appl.* 31, 4 (2010), 2146–2162.

[9] Vincent D. Blondel, Yurii Nesterov, and Jacques Theys. 2005. On the accuracy of the ellipsoid norm approximation of the joint spectral radius. *Linear Algebra Appl.* 394, 1 (2005), 91–107.

[10] Vincent D. Blondel and John N. Tsitsiklis. 1997. The Lyapunov exponent and joint spectral radius of pairs of matrices are hard – when not impossible – to compute and to approximate. *Math. Control Sign. Syst.* 10, 1 (1997), 31–40.

[11] Vincent D. Blondel and John N. Tsitsiklis. 2000. The boundedness of all products of a pair of matrices is undecidable. *Syst. Control Lett.* 41, 2 (2000), 135–140.

[12] Maria Charina and Thomas Mejstrik. 2018. Multiple multivariate subdivision schemes: matrix and operator approaches. *Comput. Appl. Math.* 349 (2018), 279–291.

[13] Maria Charina and Vladimir Yu. Protasov. 2019. Regularity of anisotropic refinable functions. *Appl. Comput. Harm. A.* 47, 3 (2019), 795–821.

[14] Ingrid Daubechies. 1988. Orthonormal bases of compactly supported wavelets. *Comm. Pure Appl. Math.* 41 (1988).

[15] Ingrid Daubechies and Jeffrey C. Lagarias. 1992. Two-scale difference equations. ii. local regularity, infinite products of matrices and fractals. *SIAM J. Math. Anal.* 23, 4 (1992), 1031–1079.

[16] Gustav Gripenberg. 1996. Computing the joint spectral radius. *Linear Alg. Appl.* 234 (1996), 43–60.

[17] Nicola Guglielmi and Vladimir Yu. Protasov. 2013. Exact Computation of Joint Spectral Characteristics of Linear Operators. *Found. Comput. Math.* 13 (2013), 37–39.

[18] Nicola Guglielmi and Vladimir Yu. Protasov. 2015. Matrix approach to the global and local regularity of wavelets. *Poincare J. Anal. Appl.* 2 (2015), 77–92.

[19] Nicola Guglielmi and Vladimir Yu. Protasov. 2016. Invariant polytopes of linear operators with applications to regularity of wavelets and of subdivisions. *SIAM J. Matrix Anal. & Appl.* 37, 1 (2016), 18–52.

[20] Nicola Guglielmi, Fabian Wirth, and Marco Zennaro. 2005. Complex polytope extremality results for families of matrices. *SIAM J. Matrix Anal. Appl.* 27, 3 (2005), 721–743.

[21] Nicola Guglielmi and Marco Zennaro. 2008. An algorithm for finding extremal polytope norms of matrix families. *Linear Alg. Appl.* 428, 10 (2008), 2265–2282.

[22] Nicola Guglielmi and Marco Zennaro. 2009. Finding extremal complex polytope norms for families of real matrices. *SIAM J. Matrix Anal. Appl.* 31, 2 (2009), 602–620.

[23] Leonid Gurvits. 1995. Stability of discrete linear inclusion. *Linear Alg. Appl.* 231 (1995), 47–85.

[24] Kevin G. Hare, Ian D. Morris, Nikita Sidorov, and Jacques Theys. 2011. An explicit counterexample to the Lagarias–Wang finiteness conjecture. *Adv. Math.* 226, 6 (2011), 4667–4701.

[25] Julien M. Hendrickx, Raphaël Jungers, and Guillaume Vankeerberghen. 2014. JSR: A Toolbox to Compute the Joint Spectral Radius. mathworks.com/matlabcentral/fileexchange/33202.

[26] Victor S. Kozyakin. 2010. Iterative building of Barabanov norms and computation of the joint spectral radius for matrix sets. *Discrete Continuous Dyn. Syst. Ser. B* 14, 1 (2010), 143–158.

[27] Claudia Möller and Ulrich Reif. 2014. A tree-based approach to joint spectral radius determination. *Linear Alg. Appl.* 463 (2014), 154–170.

[28] Bruce E. Moision, Alon Orlitsky, and Paul H. Siegel. 2001. On codes that avoid specified differences. *IEEE Trans. Inf. Theory* 47 (2001), 433–442.

[29] Pablo A. Parrilo and Ali Jadbabaie. 2008. Approximation of the joint spectral radius using sum of squares. *Linear Alg. Appl.* 428, 10 (2008), 2385–2402.

[30] Vladimir Yu. Protasov. 2000. Asymptotic behaviour of the partition function. *Sb. Math.* 191, 3–4 (2000), 230–233.

[31] Gian-Carlo Rota and Gilbert W. Strang. 1960. A note on the joint spectral radius. *Kon. Nederl. Acad. Wet. Proc.* 63 (1960), 379–381.