**Numerical Algorithms manuscript No.**
(will be inserted by the editor)

# An experimental study of approximation algorithms for the joint spectral radius

**Chia-Tche Chang · Vincent D. Blondel**

Received: date / Accepted: date

**Abstract** We describe several approximation algorithms for the joint spectral radius and compare their performance on a large number of test cases. The joint spectral radius of a set $\Sigma$ of $n \times n$ matrices is the maximal asymptotic growth rate that can be obtained by forming products of matrices from $\Sigma$. This quantity is NP-hard to compute and appears in many areas, including in system theory, combinatorics and information theory. A dozen algorithms have been proposed this last decade for approximating the joint spectral radius but little is known about their practical efficiency. We overview these approximation algorithms and classify them in three categories: approximation obtained by examining long products, by building a specific matrix norm, and by using optimization-based techniques. All these algorithms are now implemented in a (freely available) MATLAB toolbox that was released in 2011. This tool-

This article presents research results of the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office. The scientific responsibility rests with the authors. Chia-Tche Chang is a F.R.S.-FNRS Research Fellow (Belgian Fund for Scientific Research). A preliminary version of the second section of this article has been presented at the 18th IFAC World Congress, Milano, 2011 [6].

C.-T. Chang
Université catholique de Louvain
ICTEAM Institute
Department of Mathematical Engineering
avenue Georges Lemaître, 4
B-1348 Louvain-la-Neuve, Belgium
Tel.: +32-10-472264
E-mail: chia-tche.chang@uclouvain.be

V. D. Blondel
Université catholique de Louvain
ICTEAM Institute
Department of Mathematical Engineering
avenue Georges Lemaître, 4
B-1348 Louvain-la-Neuve, Belgium

box allows us to present a comparison of the approximations obtained on a large number of test cases as well as on sets of matrices taken from the literature. Finally, in our comparison we include a method, available in the toolbox, that combines different existing algorithms and that is the toolbox's default method. This default method was able to find optimal products for all test cases of dimension less than four.

**Keywords** Joint spectral radius · Generalized spectral radius · Product of matrices · Matrix semigroup · Dynamical systems · Discrete-time systems

**Mathematics Subject Classification (2000)** 15A18 · 15A60 · 65F15 · 65F35

## 1 Introduction

The *joint spectral radius* $\rho(\Sigma)$ of a set of matrices $\Sigma \subset \mathbb{R}^{n\times n}$ is a quantity that characterizes the maximal asymptotic growth rate of products of matrices drawn from the set $\Sigma$. It is formally defined by

$$\rho(\Sigma) = \lim_{t\to\infty} \rho_t(\Sigma), \tag{1}$$

with

$$\rho_t(\Sigma) = \max\left\{ \|M\|^{1/t} \mid M \in \Sigma^t \right\}.$$

Here, $\Sigma^t$ denotes the set of products of length $t$ of matrices in $\Sigma$. One can show that the limit in (1) exists and that it does not depend on the matrix norm that is used, provided this norm is submultiplicative. In the particular case where $\Sigma$ contains only one matrix, the joint spectral radius is equal to the usual spectral radius, i.e., the largest magnitude of the eigenvalues. Equation (1) is thus a generalization of the well-known Gelfand formula for the spectral radius of a single matrix.

The joint spectral radius was initially introduced by Rota and Strang in [29] and has since then appeared in many applications such as the stability of switched systems [12,18], the study of wavelets [8,24], combinatorics and language theory [15], the capacity of some types of codes [25], etc. More examples and a general overview of what is known for the joint spectral radius can be found in [14]; see also [30] for more specific results.

The problem of approximating the joint spectral radius has been widely studied. The first algorithms proposed consisted in constructing products of increasing length and using $\rho_t$ as upper bounds. Lower bounds can be obtained by considering the *generalized spectral radius* $\bar\rho(\Sigma)$ defined by

$$\bar\rho(\Sigma) = \limsup_{t\to\infty} \bar\rho_t(\Sigma), \tag{2}$$

with

$$\bar\rho_t(\Sigma) = \max\left\{ \rho(M)^{1/t} \mid M \in \Sigma^t \right\}.$$

For all lengths $t$, the following inequalities are satisfied [1]:

$$\bar{\rho}_t(\Sigma) \leq \bar{\rho}(\Sigma) \leq \rho(\Sigma) \leq \rho_t(\Sigma). \tag{3}$$

It was moreover proved in [1] that the generalized spectral radius is equal to the joint spectral radius whenever $\Sigma$ is bounded (or, in particular, finite). For bounded sets, the inequalities in (3) can thus be used to derive arbitrarily precise approximations for the joint spectral radius.

It has also been observed that the lower bound in (3) often reaches the joint spectral radius for some finite $t$. A set of matrices $\Sigma$ is said to possess the *finiteness property* if there exists some product $M \in \Sigma^t$ such that $\rho(\Sigma) = \rho(M)^{1/t}$. It was first conjectured by Lagarias and Wang in [22] that all sets of matrices do have the finiteness property — a conjecture known as the *finiteness conjecture* — but this conjecture was proved to be false [5,4,17]. The proofs in these articles are nonconstructive and recently an explicit counterexample has been provided by Hare et al. in [13] by extending the results in [5,4,18]. Note that the question of determining if all sets of *rational* matrices have the finiteness property is still open. This "restricted" finiteness conjecture is interesting because matrices that appear in applications often have rational, integer or even binary entries.

The sequence of upper bounds in (3) converges very slowly to $\rho(\Sigma)$ except in some particular cases. Hence, any approximation algorithm directly based on the inequalities (3) is bound to be inefficient. This is not so surprising since the problem of approximating the joint spectral radius is known to be NP-hard [31]: unless P=NP, there is no algorithm that, given a set of matrices $\Sigma$ and a relative accuracy $\varepsilon$, is able to return an $\varepsilon$-approximation of $\rho(\Sigma)$ in a time polynomial in the size of $\Sigma$ and $\varepsilon$.

Our contribution in this paper consists of two parts. First, we review a number of approximation methods that appear in the literature, we classify them in three categories and we summarize their main characteristics. This extensive review of the literature can be found in Section 2.

Second, we analyze the efficiency of the different methods. Many of the approximation methods described in the literature have only been analyzed from a theoretical point of view and have never been implemented, or their implementations are not publicly available. The methods that we consider in this article have all been implemented in a recent MATLAB toolbox (the *JSR Toolbox*) that is freely available for download [32]. Note that the toolbox offers a "default" method that approximates the joint spectral radius by combining different algorithms.

In Section 4, we use the JSR Toolbox to compare the performances of the different algorithms on a large number of test cases. In particular, we observe numerical issues due to finite precision arithmetic and implementation difficulties that alter the behavior of algorithms that may otherwize look promising. We discuss these issues in Section 4 and provide there final comments and recommendations.

## 2 Approximation methods for the joint spectral radius

The problem of approximating the joint spectral radius is far from trivial, except in some particular cases. A first naive approach would be to consider the relations (1), (2), (3) and evaluate all products in $\Sigma^t$ for increasing values of $t$. Of course, these bounds will converge to the exact value of the joint spectral radius, but since the number of products to consider grows exponentially with the product length, one has to reduce the search space in some way. Several arguments may be used, e.g., the fact that the spectral radius of a product of matrices is invariant under cyclic permutations of its factors [23]. This leads us to branch-and-bound based methods, which are presented in Section 2.1.

Note that the sequence of upper bounds may converge very slowly even if a lower bound may reach $\rho(\Sigma)$ for a small product due to the finiteness property. A detailed analysis of the quality of the upper bounds may by found in [19]. In order to speed up the procedure, one can also try to find an appropriate norm that gives a faster convergence rate in the expression (1). In some cases, it is even possible to find a norm that is *extremal* with respect to the set of matrices, that is, a norm such that the joint spectral radius is reached with a product of length one. More precisely, a norm $\|\cdot\|$ is said to be extremal for a set of matrices $\Sigma$ if $\|M\| \leq \rho(\Sigma)$ for all $M \in \Sigma$.

We also have the property:

$$\rho(\Sigma) = \inf_{\|\cdot\|} \sup_{M \in \Sigma} \|M\|, \tag{4}$$

where the infimum is taken over the set of all matrix norms; see also [33]. Methods for approximating the joint spectral radius with techniques based on the optimization problem (4) are presented in Section 2.2, whereas Section 2.3 describes geometric algorithms that iteratively approximate an extremal norm.

### 2.1 Product enumeration

Gripenberg's branch-and-bound method [9] was one of the first algorithms proposed for approximating the joint spectral radius. Given a target tolerance $\varepsilon$ and a norm $\|\cdot\|$, this algorithm starts with a initial set $\Pi_1 = \Sigma$ of candidates, and the corresponding natural bounds, i.e., $\alpha_1 = \max_{M \in \Sigma} \rho(M)$ as lower bound and $\beta_1 = \max_{M \in \Sigma} \|M\|$ as upper bound. In general, the upper bounds will depend on the chosen norm.

At the $k^{\text{th}}$ iteration of the algorithm, let us define the new set of candidates:

$$\Pi_k = \{MP \mid M \in \Sigma, P \in \Pi_{k-1}, \mu(MP) > \alpha_{k-1} + \varepsilon\},$$

where $\mu(M_1 \dots M_k) = \min_{1 \le i \le k} \|M_1 \dots M_i\|^{1/i}$. The bounds are then updated as follows:

$$\alpha_k = \max \left\{ \alpha_{k-1}, \max_{P \in \Pi_k} \rho(P)^{1/k} \right\},$$
$$\beta_k = \min \left\{ \beta_{k-1}, \max \left\{ \alpha_{k-1} + \varepsilon, \max_{P \in \Pi_k} \mu(P) \right\} \right\}.$$

At each iteration, the bounds satisfy $\alpha_k \le \rho(\Sigma) \le \beta_k$ and we have $\lim_{k\to\infty}(\beta_k - \alpha_k) \le \varepsilon$. Hence, this method provides arbitrarily precise approximation of the joint spectral radius $\rho(\Sigma)$ when given sufficient computational resources. The number of iterations required to achieve a given absolute accuracy of $\varepsilon$ is not known a priori and may depend on the choice of the norm $\|\cdot\|$. Even when it is possible to discard a large number of products at each step, it may be necessary to reach very long products in order to obtain an interval of length $\varepsilon$.

In [25], Moision et al. introduces a *pruning algorithm* for sets of nonnegative matrices. This method is based on several dominating relations, e.g., the fact that if $A \ge B \ge 0$ componentwise, then $\rho(A) \ge \rho(B)$. Similar conditions can also be found for the upper bounds, depending on the norm used in the algorithm. Of course this does not lead to a polynomial-time algorithm in the general case, but the number of products may be significantly reduced in several particular cases.

The main disadvantages of this family of methods are the large number of products that we may have to consider when reaching very long products, which may result in a large amount of computation time when a small interval is required, and the influence of the choice of the norm on the convergence of the sequence of upper bounds. In the general case, the sequence of upper bounds converges very slowly to the joint spectral radius. However, it may be possible to find tight lower bounds in a short amount of time as the set of matrices may have an optimal product of small length. Moreover, implementing these methods is usually straightforward as it mainly consists in manipulating products of matrices. Hence, this should not give rise to more numerical errors than basic arithmetic operations, except for one detail: if the length of the products grows too much, a direct application of the method may reach the overflow threshold, depending on the set of matrices. In order to avoid this issue, it may be wise to use some scaling of the matrices.

## 2.2 Norm optimization

Since the value of $\rho_t(\Sigma)$ converges slowly to $\rho(\Sigma)$ in the general case, another approach is try to find a so-called extremal norm in order to obtain better upper bounds. This is done by considering a modified version of (4). Indeed, optimizing on the space of all matrix norms is not an easy task, so in practice one may instead consider a subset of norms. Of course, this subset should preferably be chosen so that the restricted optimization problem is easy to solve. One such example introduced in [3] is the class of *ellipsoidal norms* (also called ellipsoid norms).

Given a positive definite matrix $P \in \mathbb{R}^{n \times n}$ (denoted by $P \succ 0$), let us consider the so-called *ellipsoidal* vector norm $\|x\|_P = \sqrt{x^T P x}$ for $x \in \mathbb{R}^n$. The associated ellipsoidal matrix norm is the corresponding induced matrix norm:

$$\|M\|_P = \max_{x \neq 0} \frac{\|Mx\|_P}{\|x\|_P} = \max_{x \neq 0} \frac{\sqrt{x^T M^T P M x}}{\sqrt{x^T P x}}. \tag{5}$$

Similarly to (4), the ellipsoidal norm approximation $\widehat{\rho}_{Ell}(\Sigma)$ is then defined by:

$$\widehat{\rho}_{Ell}(\Sigma) = \inf_{P \succ 0} \max_{M \in \Sigma} \|M\|,$$

which is obviously an upper bound on the joint spectral radius. This approximation can be easily computed using semidefinite programming (SDP). Indeed, (5) implies that for each matrix $M \in \Sigma$ we have:

$$x^T \left( \|M\|_P^2 P - M^T P M \right) x \geq 0, \qquad \forall x \in \mathbb{R}^n.$$

The ellipsoidal norm approximation corresponds thus to the minimum value of $\gamma$ such that there exists a positive definite matrix $P \succ 0$ with $\gamma^2 P - M^T P M \succeq 0$ for all matrices $M \in \Sigma$. For a given value of $\gamma$, the problem of finding a matrix $P$ corresponds to an SDP feasibility problem that may be solved efficiently, and thus the optimal value of $\gamma$ can be found by bisection.

The following bounds are known to hold for the ellipsoidal norm approximation $\widehat{\rho}_{Ell}(\Sigma)$:

$$\frac{1}{\sqrt{\eta}} \widehat{\rho}_{Ell}(\Sigma) \leq \rho(\Sigma) \leq \widehat{\rho}_{Ell}(\Sigma), \tag{6}$$

with $\eta = \min\{n, |\Sigma|\}$, where $|\Sigma|$ corresponds to the size of $\Sigma$, i.e., the number of matrices in the set. However, in practice, better lower bounds can easily be obtained by considering products of small lengths.

This SDP-based method has been generalized by several authors, including Parrilo and Jadbabaie. In [26] these authors propose a generalization by replacing the norms by positive polynomials, based on the following result:

**Proposition 1** *Let $\Sigma$ be a set of $n \times n$ matrices and $p(x)$ be a (strictly) positive homogeneous polynomial of degree $2d$ with $n$ variables that satisfies $p(Mx) \leq \gamma^{2d} p(x) \forall x \in \mathbb{R}^n$, for all $M \in \Sigma$. Then, $\rho(\Sigma) \leq \gamma$.*

Even though positive polynomials are hard to characterize, a positivity constraint can be relaxed into a sum-of-squares (SOS) constraint: instead of $p(x) \geq 0$, we require the existence of a decomposition $p(x) = \sum_i p_i(x)^2$. Note that although a sum of squares is obviously nonnegative, most but *not all* nonnegative polynomials can be rewritten as sums of squares. The sum-of-squares decomposition can also be written as $p(x) = \left(x^{[d]}\right)^T P x^{[d]}$, where $x^{[d]}$ is a vector containing all monomials of degree $d$ with $n$ variables, and $P \succeq 0$ is a positive semidefinite matrix. The problem of checking if a polynomial is a sum-of-squares is thus equivalent to a SDP feasibility problem and the sum-of-squares approximation can thus be exprimed as follows:

$$\begin{aligned}\widehat{\rho}_{SOS,2d}(\Sigma) \; = \; & \min \gamma \\ & \text{s.t. } \exists\, p(x) \in \mathbb{R}[x]_{2d} \text{ homogeneous} \\ & \quad p(x) \text{ is SOS} \\ & \quad \gamma^{2d} p(x) - p(Mx) \text{ is SOS } \forall M \in \Sigma. \end{aligned}$$

As with the ellipsoidal approximation, the optimal value of $\gamma$ can be found by bisection. In fact, in the particular case $d = 1$, the problem is equivalent to the ellipsoidal norm case because all quadratic nonnegative polynomials can be written as sums of squares.

In the general case, we have the following bounds on the approximation accuracy:

$$\left(\frac{1}{\sqrt{\eta}}\right)^{d} \widehat{\rho}_{SOS,2d}(\Sigma) \le \rho(\Sigma) \le \widehat{\rho}_{SOS,2d}(\Sigma), \tag{7}$$

where $\eta = \min\left\{\binom{n+d-1}{d}, |\Sigma|\right\}$. In theory, one can thus obtain arbitrarily sharp approximations by taking polynomials of sufficiently large degree, but the computational cost increases accordingly.

Another generalization of the ellipsoidal norm approximation was presented in [28], where the authors extend the SDP problem to general conic programming. This extension allows to derive upper and lower bounds on the joint spectral radius, provided that the matrices in $\Sigma$ leave a common cone $K$ *invariant*, i.e., $MK \subset K$ for all $M \in \Sigma$. In this case, the conic approximation $\widehat{\rho}_{Conic,K}(\Sigma)$ satisfies the inequalities $\alpha(K)\widehat{\rho}_{Conic,K}(\Sigma) \le \rho(\Sigma) \le \widehat{\rho}_{Conic,K}(\Sigma)$, where $\alpha(K) \ge \frac{1}{n}$ is a constant depending on the invariant cone $K$. For example, the constant corresponding to the cone $\mathbb{S}^n_+$ of positive semidefinite $n \times n$ matrices, has value $\alpha(\mathbb{S}^n_+) = \frac{1}{n}$. Note that for an arbitrary set $\Sigma$, it is always possible to build a set $\widetilde{\Sigma}$ leaving the cone $\mathbb{S}^n_+$ invariant and satisfying $\rho(\widetilde{\Sigma}) = \rho(\Sigma)^2$. When applying this method on the lifted set $\widetilde{\Sigma}$ with respect to the cone $\mathbb{S}^n_+$, the result obtained is equivalent to the ellipsoidal norm approximation.

The main idea of all these methods is thus to formulate an approximation of the problem in conic optimization and use efficient solving techniques to obtain the bounds on the joint spectral radius. The implementation of these methods usually makes use of an external optimization solver. The situation is roughly the opposite of product enumeration algorithms in the sense that norm optimization methods are sometimes able to find tight *upper* bounds, but are unable to provide lower bounds of good quality. However, this approach is usually inadequate for problems of large size. Indeed, the size of these optimization problems may grow very fast and computational limitations may thus become a significant issue since the optimization solver will require a large amount of memory and of computation time. Moreover with high-dimensional problems, numerical issues tend to appear during the resolution of the optimization problem by the solver.

## 2.3 Extremal norm construction

A third approach to joint spectral radius approximation is to explicitly approximate an extremal norm instead of optimizing over a chosen set of matrix norms. Several methods have been proposed by different authors, including Protasov, Guglielmi and Kozyakin. The main idea is to start with some matrix norm or, equivalently, some region corresponding to the unit ball of a norm. Iterative algorithms are then applied in order to produce an adequate sequence of norms that converges to an extremal norm. These algorithms may explicitly work with the norms, or with the corresponding unit balls. For example, [27] proposes to approximate the unit ball of an extremal norm with a sequence $(\mathcal{P}_i)_i$ of polytopes chosen in order to obtain an $\varepsilon$-approximation of the joint spectral radius after a number of iterations which can be determined a priori, depending on $\varepsilon$ and $\Sigma$. At the $i^{\text{th}}$ iteration, the new polytope $\mathcal{P}_{i+1}$ is obtained as an approximation of the convex hull of the polytopes obtained by applying the different matrices of $\Sigma$ on $\mathcal{P}_i$. Note that the computation of a convex hull may be a significant issue when dealing with high-dimensional sets of points.

Another geometric approach involving polytopes has been studied in [11] and in [7]. The method is based on balanced polytopes, which are polytopes $\mathcal{P} \subset \mathbb{K}^n$ such that there exists a finite set of vectors $\mathcal{V} = \{v_1, \dots, v_k\}$ satisfying $\operatorname{span}(\mathcal{V}) = \mathbb{K}^n$ and:

$$\mathcal{P} = \operatorname{absco}(\mathcal{V}) := \left\{ x \in \mathbb{K}^n \;\middle|\; x = \sum_{i=1}^{k} \lambda_i v_i \text{ with } \sum_{i=1}^{k} |\lambda_i| \leq 1, \lambda_i \in \mathbb{K} \right\},$$

with $\mathbb{K} = \mathbb{R}$ or $\mathbb{C}$. Here, "absco" corresponds to *absolutely convex hull.* Depending on the choice for $\mathbb{K}$, the polytope is called a *balanced real polytope* or a *balanced complex polytope.* A *polytope norm* is any norm whose unit ball is a balanced polytope. It is known [10] that the set of induced matrix polytope norms is dense in the set of induced matrix norms, so (4) holds even if we take the infimum on all polytope norms.

In order to find such a polytope norm, the authors present an algorithm that essentially considers the trajectories of a vector $\tilde{x}$ under all possible products of matrices in $\Sigma$. If $\tilde{x}$ is well-chosen and some hypotheses hold, then the convex hull of the trajectories will describe a balanced polytope, that will give the value of the joint spectral radius. In particular, it is supposed that the set $\Sigma$ do possess the finiteness property. The vector $\tilde{x}$ is then taken as a leading eigenvector of a candidate product. Note that this eigenvector may be complex even if $\Sigma \subset \mathbb{R}^{n\times n}$. The main idea of the algorithm is given below:

- Let $P$ be the candidate product (of length $m$) and let $\widetilde{\Sigma} = \rho(P)^{-1/m}\Sigma$ be a scaled version of $\Sigma$. Define the set $\mathcal{V}_0$ as $\mathcal{V}_0 = \{x, \bar{x}\}$, where $x$ is a (possibly complex) leading eigenvector, and let $\mathcal{P}_0 = \operatorname{absco}(\mathcal{V}_0)$.
- Recursively compute $\mathcal{V}_{k+1} = \widetilde{\Sigma}\mathcal{V}_k$.

– At each iteration, define $\mathcal{P}_k$ as $\mathcal{P}_k = \mathrm{absco}(\mathcal{V}_k)$. If $\mathcal{V}_{k+1} \subset \mathcal{P}_k$ for some $k$ then terminate the algorithm with the conclusion that the product $P$ is optimal.

More details can be found in [11]. This method requires a good initial guess, but if the candidate product is indeed optimal, then the algorithm stops after a finite number of steps and provides a certificate for the product optimality. In [7], a stopping criterion ($x \in \operatorname{int} \mathcal{P}_k$ for some $k$) is given so that the iteration can be stopped and the candidate product discarded if it is not optimal. Apart of the requirement of a starting point, the main drawback is that this algorithm usually requires a large amount of computation time, especially with matrices of larger sizes. Efficiently implementing the algorithm is also nontrivial (see Section 3.2). Another issue is that the operations such as the computation of the convex hulls may be subject to numerical inaccuracies or a large memory usage with high-dimensional sets of points.

Two different iterative schemes are proposed in [20] and [21] for the construction of an extremal norm. More precisely, these algorithms are mainly designed for building Barabanov norms, but they also give the joint spectral radius as a byproduct. In the first algorithm (called the *Linear relaxation iteration* or *LR-procedure*), starting with an arbitrary initial norm on $\mathbb{R}^n$, we build a sequence of norms by using linear combinations of already computed norms. Bounds on the joint spectral radius are available at every iteration. More precisely, the LR-procedure defines a sequence of norms $(\|\cdot\|_k)_{k\in\mathbb{N}}$ according to the following rules:

– Start with a norm $\|\cdot\|_0$ on $\mathbb{R}^n$ and let $e \neq 0$ be a vector such that $\|e\|_0 = 1$. Let us also choose $\lambda^-$, $\lambda^+$ such that $0 < \lambda^- \leq \lambda^+ < 1$.
– At every iteration, bounds on the joint spectral radius are given by $\rho(\Sigma) \in [\rho_k^-, \rho_k^+]$ with:

$$\rho_k^+ = \max_{x\neq 0} \frac{\max_i \|M_i x\|_k}{\|x\|_k},\ \rho_k^- = \min_{x\neq 0} \frac{\max_i \|M_i x\|_k}{\|x\|_k}.$$

– Let $\lambda_k$ be in the interval $[\lambda^-, \lambda^+]$ and define the new norm $\|\cdot\|_{k+1}$ as follows:

$$\|x\|_{k+1} = \lambda_k \|x\|_k + (1-\lambda_k) \frac{\max_i \|M_i x\|_k}{\max_i \|M_i e\|_k}.$$

This procedure converges to an extremal norm, and the two sequences $\left(\rho_k^\pm\right)_{k\in\mathbb{N}}$ converge to $\rho(\Sigma)$. Alternatively, one can also apply the *Max-relaxation iteration* that replaces the linear combination with a maximum operation and an averaging function. The MR-procedure has similar convergence properties as the LR-procedure. More details can be found in [20] and [21].

In practice, producing an efficient implementation of these two methods is quite challenging due to the geometric nature of the operations and the fact that the algorithms are described at a rather high level. One possibility would be to discretize the space but this may reduce the practical accuracy of the algorithms, due to discretization errors. Moreover, this approach can become difficult to manage with matrices of large sizes as the required number of discretization points would grow exponentially. More details about our implementation can be found in Section 3.2. This implementation seems to produce acceptable results of limited accuracy provided that the problem size is small enough.

### 2.4 Lifting techniques

Whatever method is used to approximate the joint spectral radius, it is always possible to apply a *lifting* on the set of matrices $\Sigma$. This provides better bounds at the price of a higher computational cost, for example when combined with the bounds (6) or (7). The main idea is to build a set of matrices $\widetilde{\Sigma}$ such that the relation between $\rho(\widetilde{\Sigma})$ and $\rho(\Sigma)$ is known, and then apply the algorithms on $\widetilde{\Sigma}$. Some examples are $\Sigma^t$, the set of products of length $t$, $\Sigma^{\otimes t}$, the set of $t^{\text{th}}$ Kronecker powers of the matrices in $\Sigma$, or $\Sigma^{[t]}$, the set of $t$-lifts of matrices in $\Sigma$ (see [2] and [26]). Indeed, it can be proved that

$$\rho(\Sigma)^t = \rho(\Sigma^t) = \rho(\Sigma^{\otimes t}) = \rho(\Sigma^{[t]}).$$

In these examples the lifted set contains either a larger number of matrices, or matrices that have a larger dimension. Note that in [2], the authors also propose a recursive approximation method based on successive liftings of the initial set. In theory, this may provide arbitrarily accurate bounds on the joint spectral radius, but the exponential growth of the problem size renders the method intractable except for a very small number of liftings.

## 3 Experimental analysis

In practice, the performance of all the methods that we have described varies widely. A branch-and-bound method such as Gripenberg's algorithm provides an interval containing the value of the joint spectral radius, but the computation time becomes prohibitive when a small interval is desired due to the growth of the number of products to consider. The same thing can be said about the pruning algorithm, when the matrices are nonnegative. Optimization methods based on the ellipsoidal norm or the sum-of-squares approximations mainly give an upper bound, and even if they may find the exact value in some cases, the size of the semidefinite optimization problem becomes huge when one tries to lift the matrices to improve the bounds in (6) or to increase the degree $d$ in (7). The computation time increases rapidly and numerical problems may become a significant issue. Geometric algorithms such as the LR-procedure or Protasov's method require the manipulation of geometric objects. The accuracy of the results may thus be significantly influenced by numerical issues such as discretization errors.

| Problem size | $n = 2$ | $n = 4$ | $n = 8$ |
|---|---|---|---|
| $\|\Sigma\| = 2$ | 100 sets | 100 sets | — |
| $\|\Sigma\| = 3$ | 100 sets | 100 sets | — |
| $\|\Sigma\| = 4$ | 100 sets | 100 sets | 100 sets |
| $\|\Sigma\| = 8$ | — | — | 100 sets |

**Table 1** Characteristics of the randomly generated sets of matrices. Each instance corresponds to a set $\Sigma$ of $n \times n$ matrices whose entries have been randomly chosen between $-5$ and 5 using a uniform distribution.

| | | Minimal period | | | | |
|---|---|---|---|---|---|---|
| Problem size | | [1; 2] | [3; 5] | [6; 9] | [10; 19] | 20+ |
| $n = 2$ | $\|\Sigma\| = 2$ | 90% | 5% | 2% | 1% | 2% |
| | $\|\Sigma\| = 3$ | 87% | 11% | 1% | 1% | — |
| | $\|\Sigma\| = 4$ | 77% | 20% | 3% | — | — |
| $n = 4$ | $\|\Sigma\| = 2$ | 74% | 17% | 6% | 3% | — |
| | $\|\Sigma\| = 3$ | 54% | 25% | 12% | 6% | 3% |
| | $\|\Sigma\| = 4$ | 63% | 20% | 13% | 3% | 1% |

**Table 2** Classification of the different instances according to the minimal period associated to the optimal products.

## 3.1 Details about the test sets

The different methods presented in the previous section have been tested on a large number of sets of matrices. Most examples are randomly generated matrices but the test set also includes instances obtained from applications such as the computation of the capacity of codes subject to forbidden difference patterns [25]. Indeed, the computation of a capacity may involve very large sets of matrices and/or high-dimensional matrices. More information about the random generation may be found in Table 1.

In the case of $2 \times 2$ and $4 \times 4$ matrices, an optimal periodic product reaching the exact value of the joint spectral radius has been found for every test case. As the computational cost increases with the size of the matrices, obtaining the same result for *every* $8 \times 8$ matrix in the test set was not possible due to a high computational cost. Note that all these results have been obtained using the JSR Toolbox (more details in Section 3.3). The exact value of the joint spectral radius allows us to compute the actual approximation errors for the different algorithms. The minimal period associated to the different optimal products for a given set $\Sigma$ may be considered as a rough measure of the difficulty of the problem. In Table 2, the different sets of matrices are classified according to their minimal period. Note that even with randomly generated matrices, the test set contains nontrivial instances, e.g., several sets of two $2 \times 2$ matrices with an optimal product whose length is more than 20. The smallest minimal period is 1 and the largest is 32.

## 3.2 Details about the implementations

The approximation methods have been implemented using MATLAB 7.4.0.336 (R2007a) and the computation have been performed on an Intel® Core™ 2 Quad Q9300 at 2.50 GHz with 3 GiB RAM, running Ubuntu Linux 10.04.3 LTS. The solver used for the SDP problems is SeDuMi 1.3. For Gripenberg's method, the implementation used in the tests is the one written by G. Gripenberg and available at `http://math.tkk.fi/~ggripenb/ggsoftwa.htm`.

Since the computations are performed with limited numerical accuracy, several issues may arise in practice. For example, as the upper bound in Gripenberg's method converges quite slowly in general, very long products may be required in order to reach the desired accuracy on the approximation of the joint spectral radius. However, if these products are too long, the implementation may fail due to overflow problems. This indeed happens for a small but non-negligible fraction of the test set, especially with sets containing only two small matrices. More precisely, given the same amount of computational resources, this problem is expected to happen less often for larger sets of matrices since the number of different products of a given length would be much larger. Hence, the maximal reachable product length given the same computational resources would be lower. This will be analyzed in Section 4 (see Table 3). One possible way to avoid these overflow problems is to rescale the matrices as often as required during the computation, in order to keep the growth of the matrices under control.

Norm optimization methods have been implemented as direct translations of the SDP problems described in the literature into SeDuMi format. The resolution of the optimization problem is done by SeDuMi 1.3, which is based on primal-dual interior-point methods and is freely available at `http://sedumi.ie.lehigh.edu`. When doing an iteration of the bisection method, the description of the SDP problem (in SeDuMi format) is updated instead of being recomputed. Hence, there is only a single construction of the full problem, which is done at the first iteration. The tolerance threshold of the SDP solver is left as a parameter.

For Guglielmi's balanced polytope algorithm, we mainly follow the description given in [11]: a first remark is that we do not really need to compute $\mathcal{V}_{k+1} = \widetilde{\Sigma}\mathcal{V}_k$ at each iteration: it is possible to prune vectors from the set. This is done by constructing a so-called *essential system of vertices* of $\mathcal{P}_k$, which corresponds to a minimal set of vertices generating $\mathcal{P}_k$ via absolutely convex combinations. The minimality of the set is not required, but it lets us keep a smaller set of vectors during the computation. The computation of this essential system of vertices can be done via a convex hull operation in the real case (which is done by Qhull with the function `convhulln` in MATLAB), or by solving a quadratic real program in the complex case, by separating the real and imaginary parts. Testing whether all vectors in $\mathcal{V}_{k+1}$ are inside $\mathcal{P}_k$ can be done by solving a linear program in the real case, or again a quadratic real program in the complex case, as we already have an essential system of vertices describing the polytope $\mathcal{P}_k$. In our implementation, we are using the MATLAB functions `linprog` and `quadprog` but other solvers can be used.

In general, geometric algorithms are subject to more implementation issues since most geometric operations, e.g., the computation of convex hulls, are done with limited precision. Describing objects such as polytopes or ellipsoids is possible, but representing the unit ball of an arbitrary norm may be problematic. In these cases, the geometric objects are usually approximated using some kind of discretization, for example in our implementation of the LR and MR-procedures. This may lead to two practical problems. On the one hand, as the dimension of the space increases, one would obviously need an exponentially increasing number of points, which is not always reasonable. On the other hand, one has to choose the position of the sampling points. Hence, while a uniform sampling of a circle (in a space of dimension 2) is easy to perform, the general problem of uniformly sampling an $n$-dimensional hypersphere is difficult. A possible solution is to use a random sampling with uniform distribution on the hypersphere but then, the number of points has to be increased even more in order to obtain good approximations with high probability. In our implementation, we have chosen a uniform random sampling by generating vectors using independent standard gaussian distributions for every component, then rescaling them in order to obtain unit norm vectors [16]. The different unit balls are then managed by keeping the values of the norms of the discretization points. The effects of the matrices acting on these discretization points are pre-computed before the first iteration. Finding the nearest discretization point is done via the MATLAB function `dsearchn`, which also uses Qhull. Of course in the two-dimensional case, we can simply use a uniform sampling of the circle and the problem of finding the nearest discretization point becomes trivial.

## 3.3 The JSR Toolbox

The JSR Toolbox [32] is a MATLAB toolbox that provides a large set of methods for the approximation of the joint spectral radius, but also includes several helper functions, for example for comparison or analysis purposes, and several demonstration functions. One important feature is that the toolbox offers a “default” algorithm that computes bounds on the joint spectral radius by combining several approaches presented in this article. This may be useful if one does not know which algorithm is more suitable. The behavior of this algorithm may also be parameterized as needed, for instance by setting a maximal computation time or by fine-tuning a particular step in an algorithm.

The main steps of the default algorithm are the following:

- Try to transform the problem into a set of smaller independent problems. This is possible when the matrices in the set $\Sigma$ are simultaneously block-triangularizable.

- If the matrices are nonnegative, start with the pruning algorithm in order to get some bounds $[\beta^-, \beta^+]$ on the joint spectral radius, then compute the joint conic radius, using the positive orthant as cone, and the ellipsoidal norm approximation using $[\beta^-, \beta^+]$ as initial bounds.
- If some matrices have negative entries, start with a variant of Gripenberg's algorithm in order to get some initial bounds and a candidate product. This variant may rescale the matrices during the computation in order to avoid overflows. After this first step, compute the ellipsoidal norm approximation and, if needed, try to certify optimality or to find a better candidate product using a balanced complex polytope method or a conitope method, which is a lifted polytope method.

In the next section, the results corresponding to this algorithm will also be included for comparison purposes.

## 4 Results

In order to compare the different methods, the performances of a chosen subset of methods (see Figure 1) is shown in Figures 2, 4 and 6. All these figures show the number of test cases where the algorithm was able to find a lower bound (left column) or upper bound (right column) close enough to the value of the joint spectral radius, and the average computation time required to obtain a result for a single set of matrices with the corresponding sizes.

The red squares correspond to a brute-force method, i.e., the evaluation of all products of length at most $t$, with $t \in \{2, 3, 4, 5, 6, 7, 8, 10, 12\}$, each square corresponding to one value of $t$, in this order from the left to the right. In Figure 6, values with $t > 8$ are not displayed as the computation time is then too large. The upper bounds are obtained using (3) with the standard 2-norm. Green circles are associated to Gripenberg's original branch-and-bound algorithm, using the 2-norm and a limitation on the number of evaluations of norms and eigenvalues. The maximal number of evaluations is set to a value among $\{100, 200, 500, 1000, 2000, 5000, 10000, 20000, 50000, 100000\}$. More precisely, each set of matrices has been tested with each one of these values. In the different figures, each green circle is associated to one of these values. Hence, the left-most green circle corresponds to the run with maximum 100 evaluations as it is the fastest, the second one corresponds to 200 evaluations, etc.

The ellipsoidal norm and sum-of-squares approximations (with $d \in \{1, 2, 3\}$) are represented by cyan pentagrams and magenta plusses ('+' symbol), respectively. As these techniques involve the resolution of a sequence of semidefinite optimization problems, three tolerance values have been tested for the SeDuMi part: $10^{-5}, 10^{-7}, 10^{-9}$. Indeed, this may influence the computation time and the result, for example if numerical problems are encountered. The initial bounds associated to the bisection method are obtained by taking the maximal spectral radius and the maximal 2-norm among all matrices in the set.

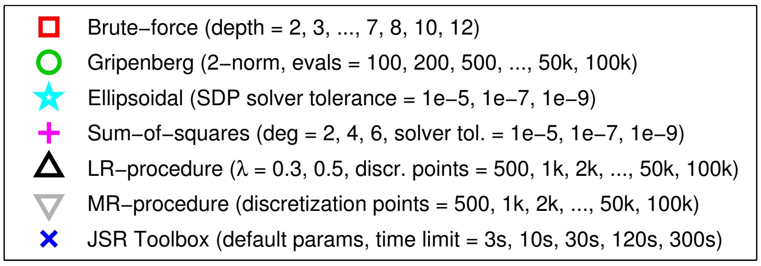

**Fig. 1** Legend associated to Figures 2, 4 and 6.

The points associated to LR and MR-procedures are displayed using black and gray triangles. The parameters $\lambda^{\pm}$ for the Linear relaxation iteration have been chosen as $\lambda^{+} = \lambda^{-} \in \{0.3, 0.5\}$. For the Max-relaxation iteration, the chosen averaging function correspond to the arithmetic mean $\mu(x, y) = \frac{x+y}{2}$. Random uniform sampling is used except for $2 \times 2$ matrices, where the natural deteministic uniform sampling of the circle is used. The algorithms have been tested with all the following values for the number of discretization points: $500, 1000, 2000, 5000, 10000, 25000, 50000, 100000$. Moreover, as the result may vary due to the random component, each run has been repeated 25 times. The corresponding results have been averaged over these 25 repetitions.

Finally, the results obtained by the JSR Toolbox are shown using blue crosses. All parameters have been kept at their default values, except the maximal allowed computation time. This limit has been set to 3, 10, 30, 120 and 300 seconds, so that bounds are obtained in a reasonable time.

Guglielmi's balanced polytope methods are not represented in these figures. Indeed, they correspond more to certification algorithms used to check whether a *given* candidate product is optimal. This means that their performance would heavily depend on the choice of this starting point, especially when we consider the amount of computation time. Nevertheless, it should be noted that such certification algorithms are implicitely used by the JSR Toolbox (see Section 3.3). For example, in our experiments, they allowed us to find the exact value of the joint spectral radius by producing optimality certificates for given products.

A first comparison of the behavior of the different methods can be seen in Figure 2. This corresponds to a set of low-dimensional test cases as we have $n = 2$ and $|\Sigma| = 2$, which can be considered as "easy". Indeed, a simple brute-force method is able to quickly reach an optimal lower bound in nearly all cases, which is consistent with Table 2. As expected, the upper bounds converge very slowly. Even with products of length 12, the error is higher than $10^{-2}$ more than 40% of the time and an accuracy of $10^{-6}$ is never reached. This can be improved by using a branch-and-bound technique, such as Gripenberg's algorithm. Indeed, the figure shows that better upper bounds are obtained, at least when the number of evaluations is low. There is no clear improvement for lower bounds as the brute-force approach was already nearly optimal. When

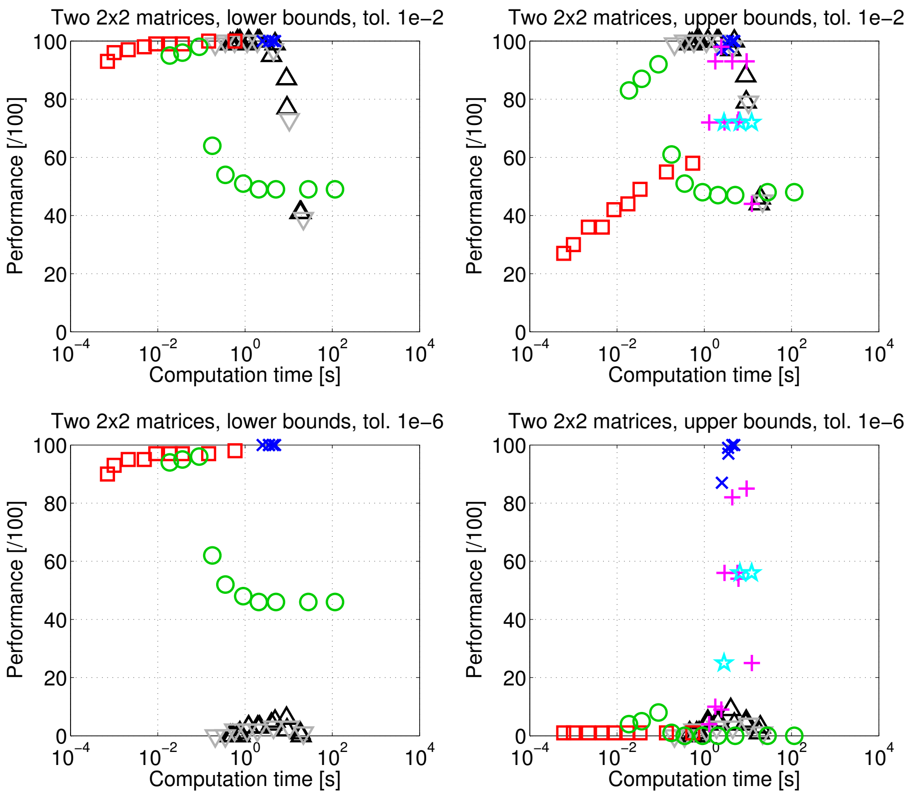

**Fig. 2** Results on small test instances (sets of two $2 \times 2$ matrices). The performance of an algorithm corresponds to the number of test cases where it was able to find a bound that is within a given tolerance ($10^{-2}$ or $10^{-6}$) of the exact value of the joint spectral radius. The computation time corresponds to the average time required by the algorithm to produce a result and is represented using a logarithmic scale.

the number of allowed evaluations is increased, the results show a significant drop in performance, which can be explained by overflow problems. Indeed, since the problem size is very small, a large number of evaluations corresponds to the evaluation of very long products, and the algorithm fails as it is unable to manage the corresponding matrices. Hence, increasing the size of the products may not be a good idea in practice *in this case* as this problem occured for a large number of instances. Fortunately, the issue is expected to happen much less often with instances of larger sizes (see Figures 4 and 6). Nevertheless, in order to avoid these numerical issues with small sets of matrices, one possible solution is to modify the algorithm by using successive rescalings of the matrices so that the products remain tractable. This is done in the branch-and-bound exploration in the first step of the JSR Toolbox. Table 3 shows the number of instances where the original version of Gripenberg's algorithm had overflow problems, for several test sets.

| Max number of evaluations | 500 | 1000 | 2000 | 5000 | 50000 |
|---|---|---|---|---|---|
| Sets of two $2 \times 2$ matrices | — | 34 | 44 | 48 | 50 |
| Sets of four $2 \times 2$ matrices | — | — | 29 | 49 | 53 |
| Sets of four $4 \times 4$ matrices | — | 4 | 8 | 11 | 12 |
| Sets of four $8 \times 8$ matrices | — | — | — | 1 | 1 |

**Table 3** Number of instances of the given sizes where Gripenberg's original algorithm had overflow problems, depending on the maximal number of evaluations. Each of the four categories of test cases contains 100 instances.

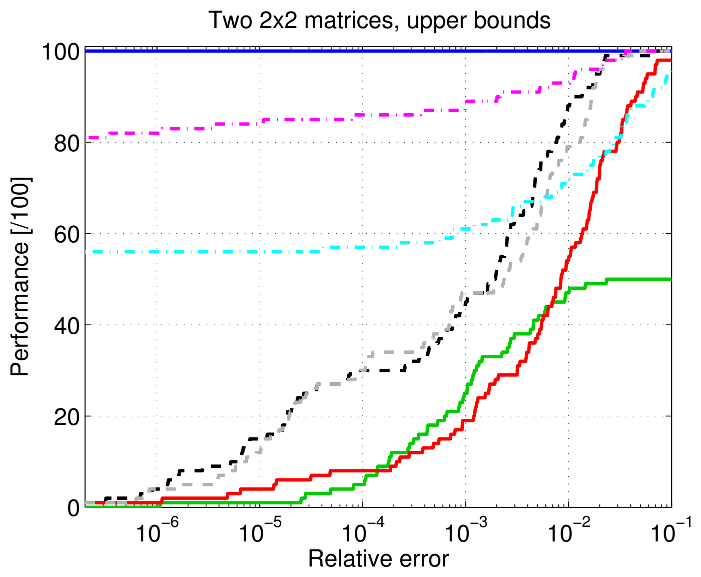

**Fig. 3** Performance of the different algorithms on sets of two $2 \times 2$ matrices, with respect to the relative error tolerance. Parameters have been chosen such that the computation times are around 7 seconds. The colors correspond to the legend in Figure 1: solid lines are associated to JSR Toolbox (blue, top), Gripenberg (green, bottom) and Brute-force (red, middle); dash-dot lines are SOS (magenta, top) and Ellipsoidal (cyan, bottom); dashed lines are LR (black) and MR (gray).

Optimization methods such as the ellipsoidal norm or the sum-of-squares approximations are able to obtain a much better accuracy, but only for a subset of all test cases. Indeed, these algorithms may directly reach the exact value of the joint spectral radius in a non-negligible number of cases, in contrast to product enumeration algorithms. This can be seen in the bottom right part of the figure as they are indeed the only methods (excluding the JSR Toolbox) that have been able to obtain upper bounds within a tolerance of $10^{-6}$. Note also that even though the sum-of-squares approximation is supposed to give better bounds than the simple ellipsoidal norm approximation, this is not always the case in practice because of the fact that increasing the degree of the polynomials also increases the chances for the optimization solver to run into numerical problems.

Comparing the results for the two tolerance values also shows that the LR and MR-procedures are able to obtain upper and lower bounds most of the time, but only with a limited accuracy due to discretization errors. Indeed, the large number of points required to reach a tolerance of $10^{-6}$ would imply a correspondingly large computation time. In order to avoid this drawback, algorithms based on successive approximations of (unit balls of) norms would have to restrict themselves to subsets of norms, e.g., polytope norms.

Finally, the JSR Toolbox seems to be able to reach the exact value in all cases for this test set. This is due to the fact that most set of matrices have a short optimal product that can be found by the branch-and-bound method. Although the upper bounds obtained by this first step are still weak, the polytope or conitope algorithm directly tries to prove the optimality of the candidate. This explains the performance in the upper bounds subfigures. Note that in these simple cases, the total computation time is a bit higher than the other algorithms with a high performance rating since with the standard parameters, three different methods are launched successively.

Figure 3 shows the quality of the upper bounds found by the algorithms when considering different tolerance values. For each algorithm, one representant has been selected. These representants have been chosen so that their computation times are roughly the same. More precisely, the computation times are all around 7 seconds in this case. This allows most algorithms to be close to their best performance (see Figure 2). The results shown in Figure 3 support the observations given in the previous paragraphs. This is expected as Figure 3 is simply another view of what happens between the two subfigures in the right column of Figure 2.

Figure 4 summarizes the results for a set of larger problems, which allows us to observe the influence of the size of the problem on the results. The counterpart of Figure 3 is also presented as Figure 5, with an average computation time of about 50 seconds. As expected, the general performance tends to be weaker, while the computation time is higher. Still, several interesting observations can be made. First, the improvements of Gripenberg's algorithm with respect to a naive brute-force approach begin to appear, at least when there are no overflow issues. The fact that its performance increases when comparing Figures 3 and 5 is due to the fact that overflows only happened in a small number of instances in this case. The upper bounds are still quite weak, but this also holds for most of the other algorithms: for example, the performance of the ellipsoidal norm approximation method is also dropping by a large factor compared to the $2 \times 2$ case. In fact, the only method that is still able to obtain tight upper bounds around the exact value with nearly all instances is the algorithm of the JSR Toolbox. Note that the success rate shown in the figure is a bit less than 100%. In fact, it is possible to reach a performance rating of exactly 100%, by increasing a little bit the threshold on the computation time. Indeed, optimal products for all 600 test sets with $n \in \{2, 4\}$ are known thanks to results obtained with this algorithm.

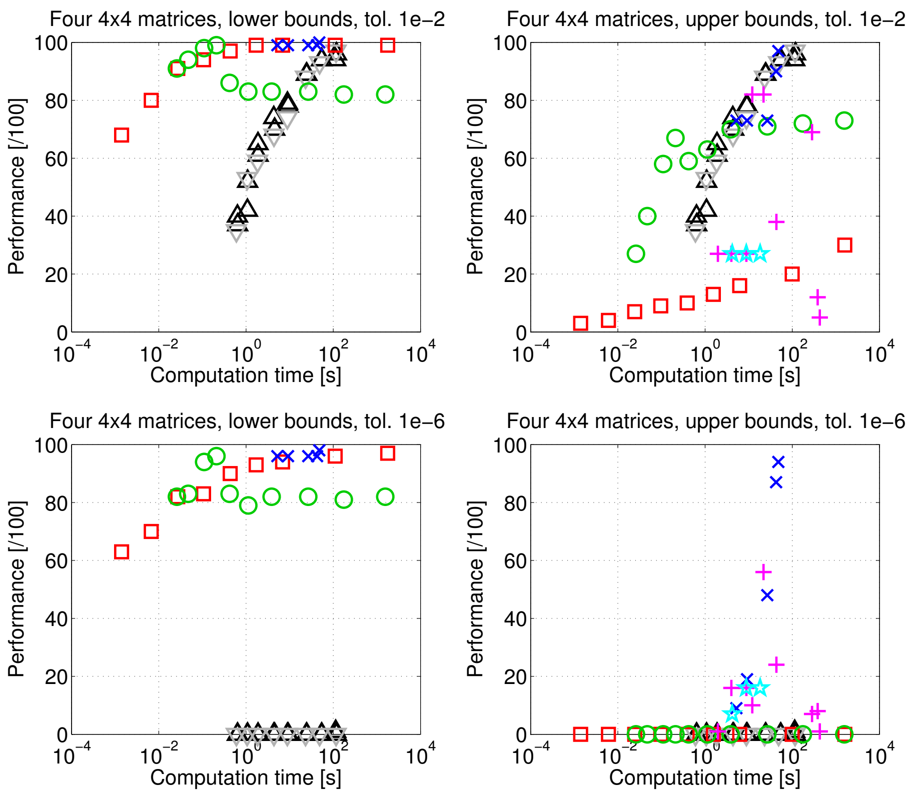

**Fig. 4** Results for sets of four $4 \times 4$ matrices. The performance measure is the same as in Figure 2.

The implementations of the LR and MR-procedures are also close to their limits as a large number of discretization points are required even for a tolerance of $10^{-2}$. Of course, the threshold at $10^{-6}$ is unreachable in a reasonable time. Note also that in practice, for an arbitrary set of matrices, it is even possible to obtain an interval that does *not* contain the actual value of the joint spectral radius due to discretization errors and thus depending on the quality of the discretization. As we need an exponentially growing number of points when the size of the problem increases, it is clear that this approach does really not scale well to high-dimensional problems.

This last remark is also valid for optimization-based methods. Indeed, Figure 4 shows that the ellipsoidal norm approximation and thus the sum-of-squares approximation of degree 2 are now inadequate in most of the test cases and even increasing the degree of the polynomial does not always yield much better results in the general case.

In Figure 6, a partial comparison between three algorithms is shown. The other algorithms are not represented as the size of the problem is now too

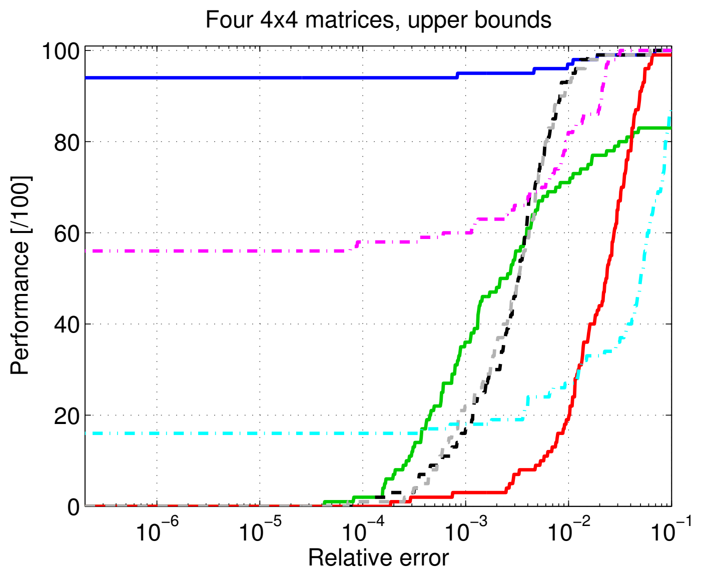

**Fig. 5** Performance of the different algorithms on sets of four $4 \times 4$ matrices, with respect to the relative error tolerance. Parameters have been chosen such that the computation times are around 50 seconds. The colors follow the legend in Figure 1: solid lines are, from the left to the right, JSR Toolbox (blue), Gripenberg (green), Brute-force (red); dash-dot lines correspond to SOS (magenta, top) and Ellipsoidal (cyan, bottom); dashed lines are LR (black) and MR (gray).

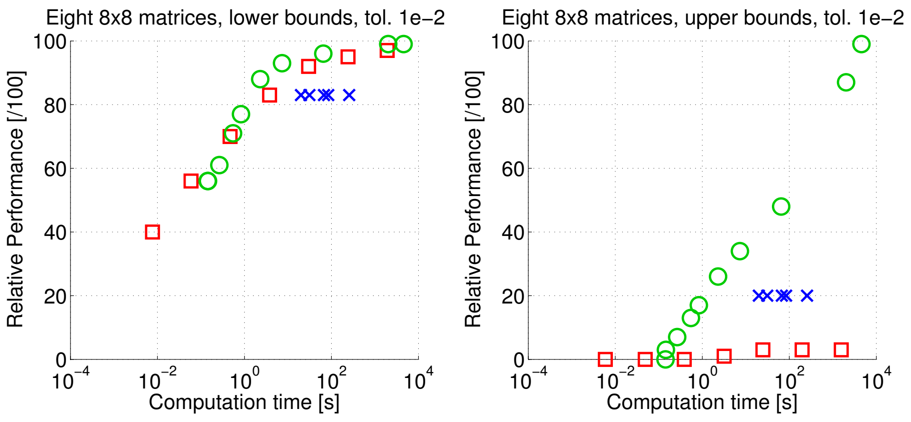

**Fig. 6** Results for large sets of eight $8 \times 8$ matrices. The exact value of the joint spectral radius is not known for all cases. The performance is therefore measured with respect to the best lower or upper bound available.

large to be able to obtain relevant results in reasonable time. Moreover, in some cases, operations such as the computation of a convex hull (which is usually done by Qhull) may become problematic. In these cases, allowing more computation time does not improve the situation. The performance is measured with respect to the best known bound because the exact value of the joint spectral radius is not available in general. Thus, a performance of 100% does *not* mean that the exact value is reached in all cases, but only that the other algorithms have not found better bounds.

Due to the large number of possible products for a set of eight matrices, Gripenberg's algorithm does not run into overflow problems and is clearly ahead of the other two methods. In this case, the JSR Toolbox did not find very good bounds. This is due to the fact that the first step is only a limited version of a variant of Gripenberg's algorithm, i.e., only short products are tested. The main part of the algorithm is the conitope step, that tries to certify the optimality of the candidate found in the first step if it is possible, while the candidate is replaced if the algorithm finds a better one. This main step is thus time-consuming, and the maximum time limit of 5 minutes is generally too low for sets of eight $8 \times 8$ matrices. Indeed, the typical behavior of the performance of this algorithm corresponds to a lower threshold where only "particularly easy" instances are completely solved, a transition region where the performance rating improves rapidly, and an upper threshold associated to "particularly hard" instances. The three situations shown in Figures 2, 4 and 6 correspond to points in the upper threshold, the transition region, and the lower threshold, respectively. Usually, the algorithm of the JSR Toolbox gives much more accurate results than Gripenberg's method starting from this transition region, especially when we consider the upper bounds. This is even more apparent in Figures 3 and 5.

## 5 Conclusion

In this article, a large set of approximation methods for the joint spectral radius has been presented. These algorithms have been implemented in MATLAB and released as part of the *JSR Toolbox*, a project available to the public via MATLAB Central. These implementations can be used to compare the performance of the different algorithms on a set of benchmarks. A default algorithm combining several approaches can also be found in the toolbox, which may be useful if one wants to approximate the joint spectral radius of a set without having to decide between the different available algorithms.

The results presented in Section 4 show that although all these methods are able to derive bounds that theoretically converge to the exact value of the joint spectral radius, this is often not observed in practice due to limitations such as numerical problems, sometimes even if a large computation time is allowed. Using a combination of several methods seems to be a good approach. For example, one can use a branch-and-bound technique to quickly find a good lower bound, and then use the corresponding product as starting point of a slower geometric algorithm. Indeed, the numerical experiments also confirm that the exact value of the joint spectral radius is often reached by a finite product, and that most of the time, such a finite product can be found in a reasonable amount of time. Of course, the main difficulty is to certify that this product is indeed optimal, but this shows the difference in difficulty between deriving good lower and upper bounds.

The comparison presented in this article can of course be useful if one wants to develop a new approximation method, for example by combining the advantages of several existing ones. Furthermore, the numerical results offer an easy way to evaluate the relative performance of new algorithms, compared to existing ones. Some algorithms may also be adapted in order to investigate other spectral quantites, such as the joint spectral subradius, which characterizes the minimal asymptotic growth rate of products of matrices, or the $p$-radius, which represents the asymptotic average growth rate of products of matrices, in the sense of the $L^p$-norm.

## References

1. Berger, M.A., Wang, Y.: Bounded semi-groups of matrices. Linear Algebra Appl. **166**, 21–27 (1992)
2. Blondel, V.D., Nesterov, Y.: Computationally Efficient Approximations of the Joint Spectral Radius. SIAM J. Matr. Anal. Appl. **27**(1), 256–272 (2005)
3. Blondel, V.D., Nesterov, Y., Theys, J.: On the accuracy of the ellipsoid norm approximation of the joint spectral radius. Linear Algebra Appl. **394**(1), 91–107 (2005)
4. Blondel, V.D., Theys, J., Vladimirov, A.A.: An Elementary Counterexample to the Finiteness Conjecture. SIAM J. Matr. Anal. Appl. **24**(4), 963–970 (2003)
5. Bousch, T., Mairesse, J.: Asymptotic height optimization for topical IFS, Tetris heaps, and the finiteness conjecture. J. AMS **15**(1), 77–111 (2002)
6. Chang, C.T., Blondel, V.D.: Approximating the joint spectral radius using a genetic algorithm framework. In: Proc. 18th IFAC World Congress, vol. 18, pp. 8681–8686. Milano, Italy (2011)
7. Cicone, A., Guglielmi, N., Serra-Capizzano, S., Zennaro, M.: Finiteness property of pairs of 2x2 sign-matrices via real extremal polytope norms. Linear Algebra Appl. **432**(2-3), 796–816 (2010)
8. Daubechies, I., Lagarias, J.C.: Two-Scale Difference Equations II. Local Regularity, Infinite Products of Matrices and Fractals. SIAM J. Math. Anal. **23**(4), 1031–1079 (1992)
9. Gripenberg, G.: Computing the joint spectral radius. Linear Algebra Appl. **234**, 43–60 (1996)
10. Guglielmi, N., Zennaro, M.: Balanced complex polytopes and related vector and matrix norms. J. Convex Anal. **14**, 729–766 (2007)
11. Guglielmi, N., Zennaro, M.: Finding extremal complex polytope norms for families of real matrices. SIAM J. Matr. Anal. Appl. **31**(2), 602–620 (2009)
12. Gurvits, L.: Stability of discrete linear inclusion. Linear Algebra Appl. **231**, 47–85 (1995)
13. Hare, K.G., Morris, I.D., Sidorov, N., Theys, J.: An explicit counterexample to the Lagarias–Wang finiteness conjecture. Advances in Mathematics **226**(6), 4667–4701 (2011)
14. Jungers, R.M.: The Joint Spectral Radius: Theory and Applications. Springer-Verlag, Berlin, Germany (2009)

15. Jungers, R.M., Protasov, V.Y., Blondel, V.D.: Overlap-free words and spectra of matrices. Theor. Comp. Sci. **410**(38), 3670–3684 (2009)
16. Knuth, D.E.: The Art of Computer Programming, Volume 2: Seminumerical Algorithms. Addison-Wesley, Reading, MA (1997)
17. Kozyakin, V.S.: Proof of a counterexample to the finiteness conjecture in the spirit of the theory of dynamical systems. Preprint 1005, Weierstraß-Institut für Angewandte Analysis und Stochastik, Berlin (2005)
18. Kozyakin, V.S.: Structure of extremal trajectories of discrete linear systems and the finiteness conjecture. Automat. Remote Control **68**(1), 174–209 (2007)
19. Kozyakin, V.S.: On accuracy of approximation of the spectral radius by the Gelfand formula. Linear Algebra Appl. **431**(11), 2134–2141 (2009)
20. Kozyakin, V.S.: A relaxation scheme for computation of the joint spectral radius of matrix sets. J. Diff. Eq. Appl. pp. 1–16 (2010)
21. Kozyakin, V.S.: Iterative building of Barabanov norms and computation of the joint spectral radius for matrix sets. Discrete and Continuous Dynamical Systems - Series B **14**(1), 143–158 (2010)
22. Lagarias, J.C., Wang, Y.: The finiteness conjecture for the generalized spectral radius of a set of matrices. Linear Algebra Appl. **214**, 17–42 (1995)
23. Maesumi, M.: An efficient lower bound for the generalized spectral radius of a set of matrices. Linear Algebra Appl. **240**, 1–7 (1996)
24. Maesumi, M.: Calculating joint spectral radius of matrices and Hölder exponent of wavelets. Approximation theory IX **2**, 1–8 (1998)
25. Moision, B.E., Orlitsky, A., Siegel, P.H.: On codes that avoid specified differences. IEEE Trans. Inf. Theory **47**(1), 433–442 (2001)
26. Parrilo, P.A., Jadbabaie, A.: Approximation of the joint spectral radius using sum of squares. Linear Algebra Appl. **428**(10), 2385–2402 (2008)
27. Protasov, V.Y.: The joint spectral radius and invariant sets of linear operators. Fundamentalnaya i prikladnaya matematika **2**(1), 205–231 (1996)
28. Protasov, V.Y., Jungers, R.M., Blondel, V.D.: Joint spectral characteristics of matrices: a conic programming approach. SIAM J. Matr. Anal. Appl. **31**(4), 2146–2162 (2010)
29. Rota, G.C., Strang, G.: A note on the joint spectral radius. Indag. Math. **22**, 379–381 (1960)
30. Shorten, R., Wirth, F., Mason, O., Wulff, K., King, C.: Stability criteria for switched and hybrid systems. SIAM Review **49**(4), 545–592 (2007)
31. Tsitsiklis, J.N., Blondel, V.D.: The Lyapunov exponent and joint spectral radius of pairs of matrices are hard — when not impossible — to compute and to approximate. Math. Cont. Sign. Syst. **10**(1), 31–40 (1997)
32. Vankeerbergen, G., Hendrickx, J., Jungers, R., Chang, C.T., Blondel, V.: The JSR Toolbox. MATLAB® Central (2011). [Software, MATLAB® toolbox]. Files available at http://www.mathworks.com/matlabcentral/fileexchange/33202-the-jsr-toolbox
33. Wirth, F.: The generalized spectral radius and extremal norms. Linear Algebra Appl. **342**(1), 17–40 (2002)