JSR: a Toolbox to Compute the Joint Spectral Radius

Guillaume Vankeerberghen*
ICTEAM, Université catholique de Louvain
4, Av. G. Lemaître
1348 Louvain-la-Neuve, Belgium
guillaume.vankeerberghen@uclouvain.be

Julien M. Hendrickx
ICTEAM, Université catholique de Louvain
4, Av. G. Lemaître
1348 Louvain-la-Neuve, Belgium
julien.hendrickx@uclouvain.be

Raphaël M. Jungers†
ICTEAM, Université catholique de Louvain
4, Av. G. Lemaître
1348 Louvain-la-Neuve, Belgium
raphael.jungers@uclouvain.be

ABSTRACT

We present a toolbox for computing the Joint Spectral Radius of a set of matrices, i.e., the maximal asymptotic growth rate of products of matrices taken in that set. The Joint Spectral Radius has a wide range of applications, including switched and hybrid systems, combinatorial words theory, or the study of wavelets. However, it is notoriously difficult to compute or approximate; it is actually uncomputable, and its approximation is NP-hard. The toolbox compiles several recent computation and approximation methods, and also contains an automatic blackbox method for inexperienced users, selecting the most appropriate methods based on an automatic study of the matrix set provided. The tool is implemented in Matlab and is freely downloadable (with documentation and demos) from Matlab Central[1].

Categories and Subject Descriptors

G.1.0.f [**Mathematics of Computing**]: Numerical Analysis—*General - Numerical algorithms*; G.4.a [**Mathematics of Computing**]: Mathematical Software—*Algorithm design and analysis*; F.2.1 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity—*Numerical algorithms and problems*

Keywords

Matlab; Joint Spectral Radius; Switched Systems; Stability

1. INTRODUCTION

The Joint Spectral Radius (JSR) of a finite set of matrices $\mathscr{M}$ is defined by

$$\rho(\mathscr{M}) = \lim_{k\to\infty} \max\{\|A_{d_1}\cdots A_{d_k}\|^{1/k} : A_{d_i} \in \mathscr{M}\}. \qquad (1)$$

It thus represents the largest asymptotic growth rate of all possible products of matrices in $\mathscr{M}$. Note that it can more generally be defined for compact (or even larger) sets $\mathscr{M}$, but we will restrict our attention to finite sets here. See [18] for a recent monograph on the topic. The JSR was introduced by Rota and Strang in 1960 [30], and has since then been the object of extensive research, partly due to its surprisingly diverse spectrum of applications.

Maybe the most natural application concerns the stability of *switched linear dynamical systems*. In these systems, the state $x(t)$ evolves as $x(t+1) = A_t x(t)$, where A_t can be any matrix in a given set $\mathscr{M}$. Such systems can be unstable for certain sequences of matrices even when every individual matrix in $\mathscr{M}$ is stable.

The evolution of the state depends of course on the particular sequence of matrices selected, but it can be easily seen that all possible trajectories converge to 0 if $\rho(\mathscr{M}) < 1$. On the other hand, one can prove that if $\rho(\mathscr{M}) \geq 1$, then there exists a sequence of matrices $A_0, A_1, A_2, \ldots$ and an initial condition $x(0)$ for which $x(t)$ does not converge to 0. Similarly, if $\rho(\mathscr{M}) > 1$ there exists a sequence of matrices and an initial condition such that $||x(t)||$ diverges [18, Corollary 1.1].

Switched systems are present in an increasing number of contexts, including systems relying on complex communication networks subject to failures and unreliabilities, which can temporarily modify the system behavior [12, 19]. They can also be used to derive bounds for more general hybrid systems. In various natural contexts, the behaviour of the system can be modeled in the following way $x(t+1) = A_{\sigma(\mathbf{x},t)}x(t)$; where $\sigma(.;.)$ takes its values in a set indexing some set of matrices $\mathscr{M}$, and may depend on x, the history of x, and on time. This is for example typically the case when analyzing multi-agent systems where the ability of agents to communicate depends on their position (see for example [5, 33] and references therein). Analyzing the behavior of these systems is often extremely challenging due to their nonlinear and sometimes discontinuous behavior. However, the JSR $\rho(\mathscr{M})$ provides an upper bound on their convergence or divergence rate.

The JSR also has very different applications in engineering and computer science. For instance, it characterizes the smoothness properties of certain wavelet functions, among which the celebrated Daubechies' finitely supported wavelets [11]. Recently, the value

*G. V. is a F.R.I.A. Fellow
†R. M. J. is a F.R.S.-FNRS Research Associate
[1]`http://www.mathworks.com/matlabcentral/fileexchange/33202-the-jsr-toolbox`. Some of the methods also require SeDuMi, `http://coral.ie.lehigh.edu/~newsedumi/` [31].

HSCC'14, April 15–17, 2014, Berlin, Germany.

ACM 978-1-4503-2732-9/14/04 ...$15.00.
http://dx.doi.org/10.1145/2562059.2562124 .

of the JSR also proved useful in combinatorics on words. Overlap-free words are words on a finite alphabet satifsying certain combinatorial properties, and an important question in their study is to characterize the rate at which the number of words of length n grows with n. It has been shown that this number could be obtained by taking the largest of all possible products (of a certain length related to n) of two matrices. By computing bounds on the JSR of these two matrices, a new approximation of the growth rate of the number of overlap-free words was obtained, outperforming previous bounds that were obtained by ad hoc methods [21]. Similar results were obtained for α-power-free words [4]. The JSR also plays a key role in the trackability of malicious agents in sensor networks [10], or the computation of the capacity of codes subject to particular constraints [24].

Unfortunately, the JSR of a set of matrices is very hard to compute or even to approximate. More specifically, the question of determining whether $\rho(\mathscr{M}) \leq 1$ is undecidable, even if $\mathscr{M}$ only consists of two matrices [9]. In addition, it is also NP-hard to approximate, even if $\mathscr{M}$ consists of two matrices whose nonzero entries are all equal to one [32]. Meaning that there is no single algorithm that will work well on all cases.

Nevertheless, several approximation methods have been proposed over the years (see Section 2), and it appears that the JSR can be computed to a satisfactory level of accuracy at an acceptable cost in many applications.

However, even if more than a dozen approximation methods are available in the literature, very little work has been done on comparing their accuracy and efficacy on given classes of problems. We also found that several of these methods were hard to obtain, so that the interested users essentially needed to re-implement them.

More importantly, implementing and using these methods, and selecting the appropriate one for a given problem, is far from trivial, and often requires some experience in the domain. As a result, it is rather difficult for researchers of other fields to test the application of the JSR to their problems. The application of the JSR to certain combinatorial word problems was for example only made possible by the collaboration between experts in the two fields [4], and many other possible applicatons of the JSR may have been left unexplored due to the lack of such collaborations.

These two issues motivated the development of the toolbox presented here, which contains two main features.
a) A compilation of fourteen recent methods for approximating the JSR in a unified interface. The goal of this part is to allow experienced users to use and test different techniques on their problems, to compare the efficiency of different methods (tuning some methods' parameters when relevant), or to help them comparing new methods that they have developed with the existing ones. Note that the toolbox will be open and that the addition of any new method by an interested contributor is welcome. Our goal is also to propose a benchmark of matrices that would be available to test new methods.
b) A ready-to-use `jsr` function with which the user can compute the JSR of a set of matrices in blackbox and that requires no understanding of the methods used. This function analyzes the set of matrices provided by the user, pre-treats them when relevant, and then selects the methods that appear the most appropriate. Our goal here is to provide researchers from other fields with a simple way of computing the JSR and to test if the results can help them in their research field.

2. DESCRIPTION OF THE METHODS

We describe here the theoretical ideas behind the methods implemented in the toolbox. The precise way to use each method in Matlab is extensively described in the `help` and the toolbox itself contains demos. A list of all implementations can be obtained in Matlab with the command `help JSR_louvain` if the toolbox's folder is named `JSR_louvain`.

Most methods to approach the JSR rely on the following inequalities, bounding it from above and from below.

$$\max\{\rho(A) : A \in \mathscr{M}^k\} \leq \rho(\mathscr{M})^k \leq \max\{||A|| : A \in \mathscr{M}^k\} \quad (2)$$

where $\mathscr{M}^k = \left\{ A_{d_1} \cdots A_{d_k} \;\middle|\; A_{d_j} \in \mathscr{M},\ j = 1, \ldots, k \right\}$.

After preprocessing steps, which we review in Section 2.1, there are essentially two families of methods. A first one consists in computing the spectral radius and/or the norm of products of k matrices of $\mathscr{M}$, and using the results to bound $\rho(\mathscr{M})$ from below and/or above using the general bounds (2). We review some of these methods in Section 2.2.

The second class of methods uses the fact that the upper bound in (2) is valid for any submultiplicative norm. They consist in building specific norms for which this upper bound is as small as possible. Some methods achieve this by an iterative procedure using the matrices of $\mathscr{M}$. Others directly build the norm using optimization techniques. We review the former in Section 2.3 and the latter in Section 2.4.

2.1 Preprocessing

2.1.1 *Joint triangularizability (`jointTriangul`)*

If the set of matrices is jointly triangularizable (i.e., if the matrices share a common nontrivial invariant subspace), then the matrices can be split into blocks so that the JSR is equal to the maximum of the JSR of the different diagonal blocks (see [18, Proposition 1.5]). To the best of our knowledge, no efficient algorithm is known to decide whether a set of matrices has a nontrivial invariant subspace. The toolbox uses a heuristic search that allows in some cases to find such a subspace if it exists.

2.1.2 *Joint triangularizability under permutation (`permTriangul`)*

Contrary to the above general triangularizability case, it is possible to decide in polynomial time whether a set of matrices is triangularizable *under a common permutation of the entries of the matrix* (that is, to decide whether a coordinate hyperplane is invariant). See [18, Lemma 3.1].

2.1.3 *The products lifting (`itMeth`)*

From Equation (2), for any natural number k, one can first compute all the products of length k of the matrices in $\mathscr{M}$, and then apply any method to compute the JSR of the new set in order to compute the JSR of the initial set. Note that for instance, computing the maximal spectral radius (or the maximal norm, for an arbitrary norm) of the set for increasing k already gives a method that asymptotically converges towards the JSR (this is straightforward from Equations (1) and (2)). This method is described in Section 2.2.1.

2.1.4 *The semidefinite lifting (`jsr_lift_semidefinite`)*

The semidefinite lifting introduced in [7] is another way of constructing a new set of matrices in the same spirit as in the previous subsection. That is, for any natural k, it builds a new set of matrices $\mathscr{M}^{[k]}$ such that

$$\rho(\mathscr{M}^{[k]}) = \rho(\mathscr{M})^{2k}.$$

Even though any method could be applied on the set $\mathscr{M}^{[k]}$, the main motivation of this lifting is that it allows for the use of a very fast method to obtain converging upper and lower bounds on the JSR, by summing all the matrices in $\mathscr{M}^{[k]}$, and computing the spectral radius of the obtained matrix [7, Theorem 5]. Since it is the principal motivation of performing the semidefinite lifting, we directly implemented it within the function `jsr_lift_semidefinite`. See also [18, Section 2.3.6].

2.1.5 *The Kronecker lifting (`itMeth`)*

The Kronecker lifting also consists in building a new set of matrices formed by taking the Kronecker product of each matrix with itself. This lifting has the same effect as taking all products of a certain length [7]. The method `itMeth` can apply the Kronecker lifting and call user-specified methods on the new set of matrices.

2.2 Enumeration based methods

2.2.1 *Brute force (`jsr_prod_bruteForce`)*

This method iteratively computes the products sets $\mathscr{M}^k$ and returns the simple upper and lower bounds provided by (2) with the euclidean norm.

2.2.2 *Pruning method for nonnegative matrices (`jsr_prod_pruningAlgorithm`)*

Thanks to the partial order implied by the positive orthant, if the matrices have nonnegative entries, it is possible to modify the brute force method described above by pruning the set of products at every step in order to save both time and space during the computations. This technique can prove extremely fast in some cases. See [6, 18, Section 2.3.3] for more explanations.

2.2.3 *Gripenberg's algorithm (`jsr_prod_Gripenberg`)*

This method, introduced in [14], is an adaptation of the brute force technique described above, which carefully prunes some of the products with the guarantee that the lower and upper bounds are still valid, up to a certain prespecified arbitrary maximal error. If this maximal error is not taken too small, this method is quite efficient, and is usually used as a first way of getting quick bounds on the JSR. Since it works by generating the products, it has the additional advantage that it provides a guess for the optimal product.

2.3 Iterative norm methods

In this section and the next one we describe methods that work on the upper bound part of Equation (2). They consist in finding a good norm so that the upper bound is as small as possible, hence providing an approximation of the JSR *even without having to compute products of matrices.* They rely on the following fundamental theorem:

THEOREM 1. *If a set of matrices $\mathscr{M}$ is irreducible (i.e. it does not have a common nontrivial invariant subspace), then there exists a norm $||\cdot||$ such that*

$$\rho(\mathscr{M}) = \max\{||A_i|| : A_i \in \mathscr{M}\}.$$

The idea of the iterative methods is to start with a candidate optimal product, an optimal product being a product $A \in \mathscr{M}^k$ for which the left-hand side inequality in (2) is tight. If the product is really optimal, then $\mathscr{M}/(\rho(A)^{1/k})$ has JSR equal to one. One can show that one can converge to a norm as in Theorem 1 by iteratively applying the matrices of $\mathscr{M}$ to the unit ball of a particular initial norm. If the candidate product was not optimal, then $\mathscr{M}/(\rho(A)^{1/k})$ has JSR larger than one, and one can detect that when iteratively applying the matrices. See for instance [16, 17] for proofs and developments.

A significant advantage of these methods is that they can end in finite time, if the upper bound provided by the norm obtained at some step is equal to the lower bound provided by the candidate optimal product A. This actually happens rather frequently, provided that one starts with a particular initial norm, which is constructed with the leading eigenvectors of the candidate optimal product. See [18, 27] for more explanation on this phenomena, and for other similar methods not implemented in this toolbox.

2.3.1 *Balanced Real Polytope method (`jsr_norm_balancedRealPolytope`)*

This method, described in [17], can be used if the leading eigenvectors of the candidate product are real. Fig. 1 illustrates a Balanced Real Polytope, i.e., a polytope with nonempty interior and symmetric around the origin, (plain polytope) such that its image by any of the matrices in $\mathscr{M}$ (dashed polytopes) is contained in the polytope. This proves that $\rho(\mathscr{M}) \leq 1$ (we do not present here the set $\mathscr{M}$ because of length constraints).

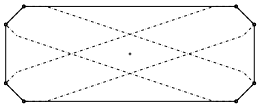


Figure 1: A Balanced real Polytope (plain lines) and its images by two matrices in $\mathbb{R}^{2\times 2}$ (dashed lines). This provides a guarantee that the JSR is smaller than or equal to one.

2.3.2 *Balanced Complex Polytope (BCP) method (`jsr_norm_balancedComplexPolytope`)*

This method, described in [15, 16] can be used in the general case where the leading eigenvectors of the candidate product have complex values.

2.3.3 *Lifted Polytope method (`jsr_norm_conitope`)*

This method, recently introduced in [20], mixes the spirit of the iterative methods with algebraic properties of liftings like the ones described above. The authors of [20] introduce new types of norms, which allow to prune vertices of the norm constructed at every step, hence allowing to save computational time and space. They also sometimes allow to deliver shorter certificates of optimality.

2.3.4 *Kozyakin's numerical algorithms (`jsr_norm_linearRelaxation2D`, `jsr_norm_maxRelaxation2D`, `jsr_norm_maxRelaxation`)*

These methods implement the numerical algorithms described in [22, 23]. In these papers, the author proposes numerical ways to approximate the norm provided by Theorem 1 and proves that they asymptotically converge towards the true value.

2.4 Optimization-based norm methods

The methods described here also try to compute a norm in order to provide a small upper bound in Equation (2). However, contrary to Section 2.3, they do not rely on iterative application of the matrices in $\mathscr{M}$. Instead, they make use of optimization methods in order to minimize this upper bound, by creating a norm *ab initio*. A significant advantage of these methods is that they come with a guarantee of accuracy: for each of them, one can compute an $\varepsilon(n)$, depending only on the dimension of the matrices, such that the upper bound ρ^* obtained with this method satisfies

$$(1-\varepsilon)\rho^* \leq \rho \leq \rho^*.$$

2.4.1 *The linear norms for nonnegative matrices (*`jsr_conic_linear`*)*

If the matrices have nonnegative entries, norms yielding a bound ρ^* arbitrarily close to ρ can be obtained as the solution of linear optimization problems. See [29] for more details and an expression of $\varepsilon(n)$.

2.4.2 *The ellipsoid method (*`jsr_conic_ellipsoid`*)*

Thanks to Semidefinite Programming techniques, one can optimize on the set of ellipsoidal norms in order to minimize the upper bound ρ^*. Fig. 2 illustrates this for 2 matrices in $\mathbb{R}^{2\times 2}$. In Fig. 2, the plain ellipse corresponds to the unit ball of an ellipsoidal norm such that its image by any of the two matrices is contained in the unit ball. This provides a guarantee that the JSR is smaller than or equal to one. Upper-bounds other than one can be found by scaling the matrices. See [2, 8] for more details and an expression of $\varepsilon(n)$.

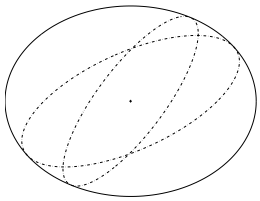


Figure 2: The ellipse corresponding to the unit ball of a norm (plain line) and its images by two matrices in $\mathbb{R}^{2\times 2}$ (dashed lines). This provides a guarantee that the JSR is smaller than or equal to one.

2.4.3 *Sum-Of-Squares (*`jsr_opti_sos`*)*

It has been shown in [26] that the idea of the ellipsoidal norm can be pushed further, by replacing ellipsoidal norms (represented by Sum-of-squares of degree 2) with higher degree Sum-Of-Squares functions in order to improve the accuracy of the bound. See [18, 26, Section 2.3.7] for more details and an expression of $\varepsilon(n)$.

3. BLACKBOX METHOD: `JSR`

In the previous section we introduced and briefly described the different algorithms implemented in the JSR Toolbox. As explained in the introduction however, selecting the best method to use for a given problem is not necessarily easy and may require some experience in the domain. As a result, a user unfamiliar with these methods might find them difficult to use. Therefore, we have included in the toolbox an easily callable function that automatically checks conditions on the set of matrices and then sequentially runs appropriate methods with the goal of providing as good as possible bounds in a reasonable time. This function is named `jsr`. Our goal is to allow anyone to check the approximate value of the JSR of a set of matrices, without requiring her or him to know anything about the computation methods used. The implementation is assorted with explanatory outputs that are understandable by non-experts. See Fig. 3 for a flow-chart of the algorithm.

The algorithm first checks if the matrices are real and nonnegative. Indeed, many applications lead to nonnegative matrices [4, 10, 21, 24], and it turns out that specific and more efficient methods can be applied in that case. Then, `jsr` checks if the matrices can be jointly block-triangularized. The method first checks for block-triangularization by permutation with `permTriangul` and if it did not work it tries with the heuristic `jointTriangul`. If none is found, it will assume that there is no joint block-triangularization. In this case it can be proved that the methods called by `jsr` will still converge to the true value, even if the matrices are in fact block-triangularizable [20]. Depending on the results of this test, it selects one of the three options described below.

Joint block-triangularizable matrices

As mentioned in Section 2.1.1, when matrices can be block-triangularized, the JSR of the set is equal to the maximum of the joint spectral radii of the sets of diagonal blocks [18]. Hence in this case, the computation of the JSR can be done by computing the JSR of each set of blocks and taking the largest one, which can usually be done at a smaller cost. Because the JSR of the set is the largest JSR of the sets of blocks, it is not always necessary to compute precisely the JSR of every block: we can ignore all those for which we know that their JSR will not be the largest one. Hence, when a block-triangularization could be found, `jsr` computes quick bounds with the pruning method in the nonnegative real case and with brute force method in the case with complex or negative and positive entries. It then checks if the upper bound on some set is lower than the lower bound on some other set so as to stop the analysis on the former. After this quick pruning attempt, `jsr` calls itself on each remaining set of blocks. As these blocks have a smaller size than the original matrices these recursive calls always stop. When the recursive `jsr` calls have provided bounds on each set of blocks, `jsr` tries to prune sets of blocks in the same way as described above. If only one set of blocks remains it outputs the bounds on this set which also bound the JSR of the original set of matrices. If more than one set of blocks remain, `jsr` prints a message explaining the situation and outputs the different blocks and the best bounds found on each of them to allow the user to pursue the analysis.

Nonnegative matrices with no joint block-triangularization

For real nonnegative matrices that cannot be block-triangularized, `jsr` first launches the pruning method. Then the bounds obtained are successively refined by the linear norm and the ellipsoid methods. Finally, the iterative lifted polytope method is launched with the product attaining the pruning method's lower bound as candidate optimal product. The rationale for applying these methods is as follows. First, the pruning method on nonnegative matrices iteratively generates long products by keeping only the interesting ones, so it has a chance of finding an optimal product. Then, the linear and ellipsoid norm methods might refine the upper-bound found. Finally, lifted polytope might provide very tight bounds if the candidate optimal product is good and if it has enough time remaining.

General matrices with no joint block-triangularization found

For matrices with entries that are complex or of both signs and that could not be block-triangularized, `jsr` starts by using Gripenberg's method. This method is good to generate products of long length by keeping only the interesting ones. Then, `jsr` launches the ellipsoid method to try refining the upper-bound from Gripenberg. Finally, `jsr` launches the lifted polytope method with as input candidate optimal product the candidate optimal product output by Gripenberg. This method has the potential to find tight bounds when it is initialized with a good candidate optimal product and is given enough time.

Additional options of `jsr`

There are three options that can be specified to `jsr`:
Computation time: One can specify an approximate time-limit on the execution. In the implementation of `jsr`, the desired time-

limit is divided in smaller time-limits that are given, to the various methods used. Note that the time cannot be checked after each operation and it is hard to predict the time a future iteration will take. Therefore, the desired time-limit is not always strictly respected. By default this time-limit is set to 120 seconds.

Tolerance: The tolerance fed as option to the methods called. Its precise implication depends on the particular methods called. We refer the reader to the `help` of these methods. By default it is set to 10^{-6}.

Block-triangularization (boolean): This boolean variable allows disabling the tests for block-triangularization. There are indeed situations where it is preferable not to launch the `jointTriangul` heuristic which could take too much time, such as for instance, when dealing with many big matrices with positive and negative entries. By default, `jsr` will try to block-triangularize as described in Fig. 3.

4. EXAMPLES

4.1 Random matrices

As a first example, we use the toolbox to compare two methods on a set of 3 random matrices in $[0,1]^{10\times 10}$. The two methods are `jsr_prod_pruningAlgorithm` and `jsr_norm_conitope`. The bounds found at each depth, or product length, by these methods is represented on Fig. 4. We see that for a comparable length of products the conitope algorithm gives substantially tighter bounds. However, this comes at the expense of time as the pruning algorithm took around 3 seconds whereas the conitope one took around 2 minutes on our personal computer. The candidate product with highest growth-rate output by these two methods is the same and is A_2A_3.

The matrices, the plot of Fig. 4, and more outputs can be generated by the script `demo2_JSR.m` in the toolbox.

4.2 Wavelet

The function `waveletMat` in the toolbox generates the set of two matrices corresponding to the Nth Daubechies wavelet, for N up to 19. We launched `jsr` on the two matrices in $\mathbb{R}^{7\times 7}$ corresponding to the wavelets of index $N = 15$. The method computes upper and lower bounds in less than 2 minutes, the lower bound found is the actual value of the JSR, 0.14754035... [14], and the upper bound is $2.171 \cdot 10^{-5}$ more.

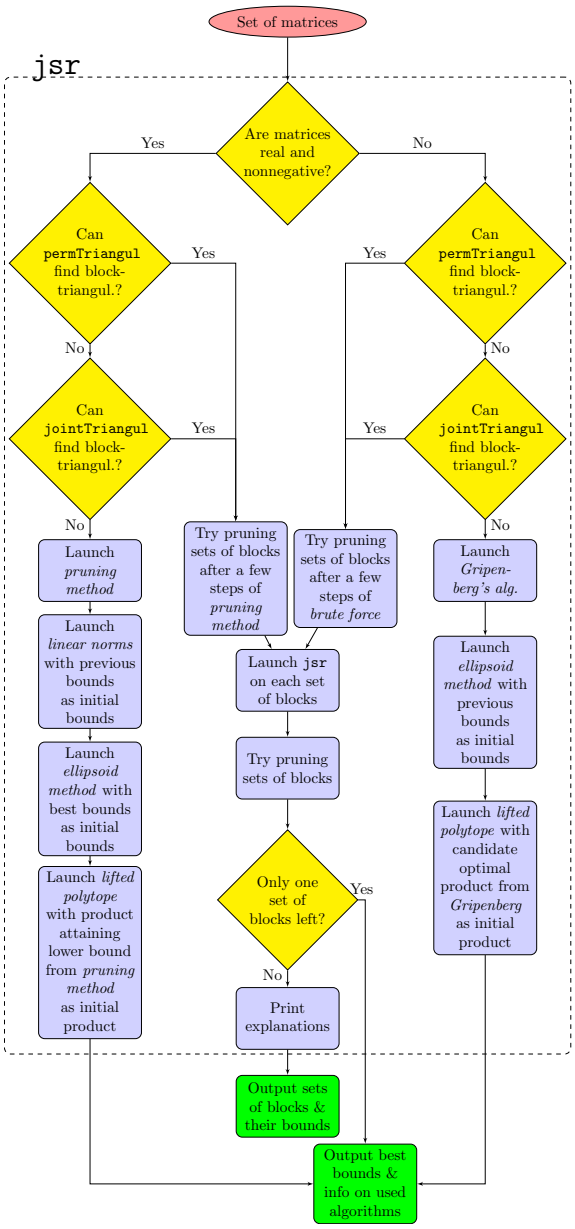
5. CONCLUSION

The JSR toolbox has been implemented and released in beta version in 2011 with two goals: to provide the practitioner with quick and efficient tools to approximate the JSR of a set of matrices, a task that can be cumbersome and sometimes necessitates to delve into quite an intricate theory; and to provide researchers with benchmarks and solid implementations of most available methods in the literature in order to compare them, adapt them to particular cases, or create new methods. We are confident that the currently available version is sufficiently solid to achieve these goals.

In the future, we plan to implement newly proposed methods that appear to outperform previous ones in some particular cases, like the so-called path-dependent Lyapunov functions recently proposed in [1]. Another development would be to broaden the scope of the toolbox towards more general tools to analyze switched systems, like the top Lyapunov exponent of matrices (see [13,28]), the joint spectral subradius [3,29], the p-radius [25], etc.

6. ACKNOWLEDGMENTS

Figure 3: Flow chart of the jsr algorithm.



We are grateful to Chia-Tche Chang and Vincent Blondel for providing their own codes for certain methods, which served as a basis for the implementation of some of the algorithms contained in the Toolbox. The research was funded by the Belgian Network DYSCO initiated by the Belgian State and by an ARC of the French Community of Belgium.

7. REFERENCES

[1] A. A. Ahmadi, R. M. Jungers, P. A. Parrilo, and M. Roozbehani. Analysis of the joint spectral radius via lyapunov functions on path-complete graphs. In *Proceedings of HSCC '11*, pages 13–22, 2011.

[2] T. Ando and M.-H. Shih. Simultaneous contractibility. *SIAM Journal on Matrix Analysis and Applications*, 19(2):487–498, 1998.

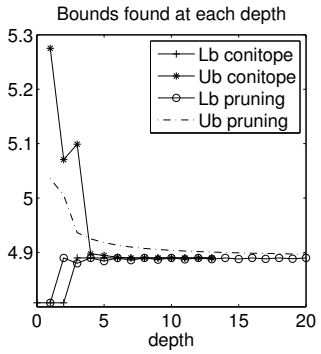


Figure 4: Bounds on the JSR found at each depth (i.e., product length) by `jsr_prod_pruningAlgorithm` and `jsr_norm_conitope`.

[3] F. Blanchini, P. Colaneri, and M.E. Valcher. Co-positive lyapunov functions for the stabilization of positive switched systems. *Automatic Control, IEEE Transactions on*, 57(12):3038–3050, 2012.

[4] V. D. Blondel, J. Cassaigne, and R. M. Jungers. On the number of α-power-free words for $2 < \alpha < 7/3$. *Theoretical Computer Science*, 410(38-40):2823–2833, 2009.

[5] V. D. Blondel, J. M. Hendrickx, and J. N. Tsitsiklis. On Krause's multi-agent consensus model with state-dependent connectivity. *Automatic Control, IEEE Transactions on*, 54(11):2586–2597, 2009.

[6] V. D. Blondel, R. M. Jungers, and V. Y. Protasov. On the complexity of computing the capacity of codes that avoid forbidden difference patterns. *Information Theory, IEEE Transactions on*, 52(11):5122–5127, 2006.

[7] V. D. Blondel and Y. Nesterov. Computationally efficient approximations of the joint spectral radius. *SIAM Journal on Matrix Analysis and Applications*, 27(1):256–272, 2005.

[8] V. D. Blondel, Y. Nesterov, and J. Theys. Approximations of the Rate of Growth of Switched Linear Systems. In *Proceedings of HSCC '04*, pages 173–186, 2004.

[9] V. D. Blondel and J. N. Tsitsiklis. The boundedness of all products of a pair of matrices is undecidable. *Systems & Control Letters*, 41(2):135–140, 2000.

[10] V. Crespi, G. Cybenko, and G. Jiang. The theory of trackability with applications to sensor networks. *ACM Transactions on Sensor Networks*, 4(3):1–42, 2008.

[11] I. Daubechies and J. C. Lagarias. Two-scale difference equations. ii. local regularity, infinite products of matrices and fractals. *SIAM Journal of Mathematical Analysis*, 23:1031–1079, 1992.

[12] P. Frasca and J. M. Hendrickx. On the mean square error of randomized averaging algorithms. *to appear in Automatica*, 2013.

[13] R. Gharavi and V. Anantharam. An upper bound for the largest lyapunov exponent of a markovian product of nonnegative matrices. *Theoretical Computer Science*, 332:543–557, 2005.

[14] G. Gripenberg. Computing the joint spectral radius. *Linear Algebra and its Applications*, 234:43–60, 1996.

[15] N. Guglielmi and V. Y. Protasov. Exact computation of joint spectral characteristics of linear operators. *Foundations of Computational Mathematics*, 13(1):37–97, 2013.

[16] N. Guglielmi, F. Wirth, and M. Zennaro. Complex polytope extremality results for families of matrices. *SIAM Journal on Matrix Analysis and Applications*, 27(3):721–743, 2005.

[17] N. Guglielmi and M. Zennaro. Finding extremal complex polytope norms for families of real matrices. *SIAM Journal on Matrix Analysis and Applications*, 31(2):602–620, 2009.

[18] R. M. Jungers. The joint spectral radius, Theory and applications. In *Lecture Notes in Control and Information Sciences*, volume 385. Springer-Verlag, Berlin, 2009.

[19] R. M. Jungers, A. D'Innocenzo, and M. D. Di Benedetto. Feedback stabilization of dynamical systems with switched delays. *Proceedings of the IEEE CDC2012*, 2012.

[20] R. M. Jungers, N. Guglielmi, and A. Cicone. Lifted polytope methods for computing the joint spectral radius. *Preprint*.

[21] R. M. Jungers, V. Y. Protasov, and V. D. Blondel. Overlap-free words and spectra of matrices. *Theoretical Computer Science*, 410:3670–3684, 2009.

[22] V. Kozyakin. Iterative building of barabanov norms and computation of the joint spectral radius for matrix. *Discrete and Continuous Dynamical Systems-series B*, 14(1):143–158, 2010.

[23] V. Kozyakin. A relaxation scheme for computation of the joint spectral radius of matrix sets. *Journal of Difference Equations and Applications*, 17(2):185–201, 2011.

[24] B. E. Moision, A. Orlitsky, and P. H. Siegel. On codes that avoid specified differences. *Information Theory, IEEE Transactions on*, 47:433–442, 2001.

[25] M. Ogura and C. F. Martin. Stability of switching systems and generalized joint spectral radius. *Proceedings of the IEEE ECC 2013*, 2013.

[26] P. A. Parrilo and A. Jadbabaie. Approximation of the joint spectral radius using Sum Of Squares. *Linear Algebra and its Applications*, 428(10):2385–2402, 2008.

[27] V. Y. Protasov. The geometric approach for computing the joint spectral radius. In *Proceedings of the 44th IEEE CDC-ECC*, pages 3001–3006, 2005.

[28] V. Y. Protasov and R. M. Jungers. Lower and Upper Bounds for the Largest Lyapunov Exponent of matrices. *To appear in: Linear Algebra and its Applications*, 2013.

[29] V. Y. Protasov, R. M. Jungers, and V. D. Blondel. Joint spectral characteristics of matrices: a conic programming approach. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2146–2162, 2010.

[30] G. C. Rota and W. G. Strang. A note on the joint spectral radius. *Indag. Math.*, 22:379–381, 1960.

[31] J.F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11–12:625–653, 1999.

[32] J. N. Tsitsiklis and V. D. Blondel. The Lyapunov exponent and joint spectral radius of pairs of matrices are hard - when not impossible - to compute and to approximate. *Mathematics of Control, Signals, and Systems*, 10:31–40, 1997.

[33] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet. Novel type of phase transition in a system of self-driven particles. *Physical Review Letters*, 75(6):1226, 1995.