MASTERARBEIT | MASTER'S THESIS

Titel | Title

Termination results for hybrid approach of Joint Spectral Radius computation

verfasst von | submitted by

Aaron Pumm, BA

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master Zusatz (Abkürzung)

Wien, | Vienna, 2025

Studienkennzahl lt. Studienblatt | degree programme code as it appears on the student record sheet:	UA 066 821
Studienrichtung lt. Studienblatt | degree programme as it appears on the student record sheet:	Masterstudium Mathematik
Betreut von | Supervisor:	Assoz. Prof. Dr. Martin Ehler

Contents

Zusammenfassung

Diese Arbeit präsentiert einen hybriden Ansatz zur Berechnung des gemeinsamen Spektralradius (JSR) einer endlichen Menge von Matrizen. Die vorgeschlagene Methode integriert die invariant-polytope und finite-tree Algorithmen, um deren jeweilige Stärken zu nutzen. Die breite Anwendbarkeit des baumbasierten Ansatzes wird mit der Effizienz der Polytop-basierten Techniken kombiniert. Der entwickelte Algorithmus konstruiert eine strukturierte Baumdarstellung und überprüft maximale Produkte mithilfe angepasster Normen, wodurch eine Terminierung in Fällen ermöglicht wird, in denen klassische Methoden an ihre Grenzen stoßen (Bedingungen an Terminierung oder Laufzeit). Theoretische Garantien werden bereitgestellt, um Korrektheit und Effizienz zu belegen. Es werden auch JSR-Berechnungen für Matrizenmengen, bei denen bestehende Methoden scheitern oder ineffizient sind demonstriert.

Abstract

This thesis presents a hybrid approach for computing the joint spectral radius (JSR) of a finite set of matrices. The proposed method integrates the invariant polytope algorithm and the finite tree algorithm to leverage their respective strengths — integrating the vast applicability of the finite-tree-based approach with the efficiency of polytope-based techniques. The developed algorithm constructs a structured tree representation and verifies spectrum-maximizing products using adapted norms, enabling termination in cases where classical methods take too long or fail. Theoretical guarantees are provided to demonstrate correctness and efficiency.

Keywords: Joint spectral radius, hybrid algorithm, invariant polytope, finite tree method, matrix norms.

1 Introduction

The *Joint Spectral Radius* (JSR) was first introduced by G.-C. Rota and G. Strang in 1960 (Rota and Gilbert Strang, 1960). They described the JSR as the maximal exponential growth rate of a product of matrices taken from a finite set. Since its inception, the JSR has become a cornerstone in various mathematical and engineering disciplines due to its ability to encapsulate the asymptotic behavior of matrix long products.

The concept gained significant traction in the 1990s when researchers began exploring its theoretical properties and practical implications. Notable advancements include its application in wavelet theory, where it assists in the construction of refinable functions (citepdaubechies1992ten) as well as in control theory, where it is used to analyze the stability of switched linear systems (Blondel and Tsitsiklis, 2000). The computational challenges associated with determining the JSR have inspired the development of several algorithms, such as the invariant-polytope method (Guglielmi and Protasov, 2013) and the finite-tree method (Möller and Reif, 2014).

Despite the progress, the JSR computation remains a challenging problem, particularly due to the exponential complexity of exploring all possible matrix products. This thesis seeks to contribute to this ongoing effort by leveraging the invariant-polytope algorithm and the finite-tree algorithm to create a hybrid methodology that mitigates their respective limitations.

To fully grasp the subsequent mathematical framework, the reader should be familiar with linear algebra, specifically matrix norms, eigenvalues, and spectral radius. A basic understanding of combinatorial optimization and algorithm design will also be beneficial.

1.1 Motivation of the JSR

[include linear switched dynamical system example]

Structure of the Thesis

The remainder of this thesis is structured as follows: Chapter 1 provides a sufficient background on the JSR and its basic properties. Chapters 2 and 3 present the ideas and concepts of the algorithms that will be exploited to form the proposed hybrid approach, outlining their theoretical foundation and algorithmic implementation. Chapter 4 discusses possible combinations of former approaches, proposes the so-called Tree-flavored-invariant-polytope algorithm, and brings proofs of termination which is the main result of this thesis. *Chapter 5* presents numerical results on ... problems to analyze the efficiency and applicability of the hybrid algorithm. Chapter 6 concludes with insights and future directions.

1.2 Theoretical background

The *joint spectral radius* (JSR) of a set of matrices is a generalization of the spectral radius for a single matrix. For a finite set of matrices $\mathcal{A} = \{A_1, A_2, \ldots, A_m\}$, the JSR is defined as:

$$\mathrm{JSR}(\mathcal{A}) = \lim_{k\to\infty} \max_{A_i \in \mathcal{A}} \|A_{i_k} \ldots A_{i_1}\|^{1/k}, \tag{1.1}$$

where $\|\cdot\|$ denotes any submultiplicative matrix norm.

Theorem 1.2.1. *The JSR is well-defined and independent of the choice of the submultiplicative norm.*

Proof. well-definedness
Let $\|\cdot\|_1$ and $\|\cdot\|_2$ be two submultiplicative norms on $\mathbb{R}^{n\times n}$. By equivalence of norms in finite-dimensional vector spaces, there exist constants $c, C > 0$ such that:

$$c\|P\|_1 \le \|P\|_2 \le C\|P\|_1 \quad \forall P \in \mathbb{R}^{n\times n}$$

Now if we consider this we get:

$$\begin{aligned}
&\lim_{k\to\infty} \max_{A_i \in \mathcal{A}} \|A_{i_k} \ldots A_{i_1}\|_1^{1/k} \\
&\le \lim_{k\to\infty} (\frac{1}{c})^{\frac{1}{k}} \max_{A_i \in \mathcal{A}} \|A_{i_k} \ldots A_{i_1}\|_2^{1/k} \\
&= \lim_{k\to\infty} \max_{A_i \in \mathcal{A}} \|A_{i_k} \ldots A_{i_1}\|_2^{1/k} \\
&\le \lim_{k\to\infty} C^{\frac{1}{k}} \max_{A_i \in \mathcal{A}} \|A_{i_k} \ldots A_{i_1}\|_1^{1/k} \\
&= \lim_{k\to\infty} \max_{A_i \in \mathcal{A}} \|A_{i_k} \ldots A_{i_1}\|_1^{1/k}
\end{aligned}$$

□

To understand the state-of-the-art algorithms considered as well as the main results that will follow, it is necessary to introduce some basic properties of the JSR.

Homogeneity

For any scalar α and set of matrices $\mathcal{A}$, the scaling property

$$\mathrm{JSR}(\alpha\mathcal{A}) = |\alpha|\,\mathrm{JSR}(\mathcal{A}) \tag{1.2}$$

holds, which follows directly from norm homogenity.

Irreducibility

Definition 1.2.2. *A set of matrices is called* (commonly) reducible *if there exists a nontrivial subspace of* $\mathbb{R}^n$ *that is invariant under all matrices in the set. This means there exists a change of basis that block-triangularizes all matrices in* $\mathcal{A}$ *at the same time. If* $\mathcal{A}$ *is not reducible it is called irreducible.*

Three-member inequality

The *three-member inequality* provides essential bounds for the JSR. For any submultiplicative matrix norm $\|\cdot\|$, the inequality

$$\max_{P\in\mathcal{A}^k} \rho(P)^{\frac{1}{k}} \leq JSR(\mathcal{A}) \leq \max_{P\in\mathcal{A}^k} \|P\|^{\frac{1}{k}}, \tag{1.3}$$

holds for every $k \in \mathbb{N}$ (Jungers, 2009). This result forms a starting point for many computational approaches as the bounds are sharp in the sense that both sides converge to the JSR as $k \to \infty$ (left side in $\limsup$).

Minimum over norms

The JSR can be equivalently defined as the minimum over all submultiplicative norms:

$$\mathrm{JSR}(\mathcal{A}) = \min_{\|\cdot\|} \max_{A\in\mathcal{A}} \|A\|. \tag{1.4}$$

Proof. blondel

□

Finiteness Property

Certain matrix sets exhibit the *finiteness property*, which states, that there exists a finite sequence of matrices $A_{i_1}, \ldots, A_{i_k}$, such that:

$$\mathrm{JSR}(\mathcal{A}) = \|A_{i_k} \cdots A_{i_1}\|^{1/k}. \tag{1.5}$$

While this property does not hold universally, it is essential for algorithmic approaches and termination in most cases.

Complexity

Candidates and Generators

Approximating the JSR requires identifying candidate products or *generators* of the matrix set that contribute most significantly to the asymptotic growth rate. These generators are often derived through optimization techniques and their identification is a key step in computational algorithms.

Similarity and reducibility

For any invertable matrix T and reducible set $\mathcal{A}$

$$\mathrm{JSR}(\mathcal{A}) = \mathrm{JSR}(T^{-1}\mathcal{A}T) \tag{1.6}$$

holds. Per definition, there exists a change of basis such that all $A \in \mathcal{A}$ are block-triangularized:

$$\exists T : T^{-1}A_iT = \begin{bmatrix} B_i & C_i \\ 0 & D_i \end{bmatrix}$$

Now we can split the set into two sets of blocks B_i and D_i and the JSR can be computed as:

$$\mathrm{JSR}(\mathcal{A}) = \max\{\mathrm{JSR}(\{B_i : i = 1, \cdots J\}), \mathrm{JSR}(\{D_i : i = 1, \cdots J\})\} \tag{1.7}$$

The proof can be seen in (Jungers, 2009). This can be applied iteratively until the sets of blocks are all irreducible. The problem was split into similar problems of smaller dimension. For the following considerations we can now assume $\mathcal{A}$ to be irreducible.

1.3 Preprocessing

This thesis aims to address the challenge of computing the JSR by combining two existing algorithms that have demonstrated practical effectiveness in calculationg the JSR for nontrivial sets of matrices. Both algorithms are based on the following simple concept:

We want to find the JSR of the finite set of matrices $\mathcal{A} = \{A_1, \cdots, A_n\}$

1. **Assumptions**: $\mathcal{A}$ is irreducible and posesses the finiteness property.
2. **Candidates**: Efficiently find products $P = A_{i_k} \cdots A_{i_1}$ of matrices from $\mathcal{A}$ that maximize the averaged-spectral radius $\hat{\rho} := \rho(P)^{\frac{1}{k}}$ for a given maximal length k_{max}.
3. **Rescaling**: Transform $\mathcal{A} \to \tilde{\mathcal{A}}$ with $\tilde{A}_i := \frac{1}{\hat{\rho}}A_i$.
4. **Proofing**: Now establish the fact that JSR$(\tilde{\mathcal{A}}) = 1$ using the three-member-inequality. By homogenity this is equivalent to JSR$(\mathcal{A}) = \hat{\rho}$.

The considered algorithms only differ in step 4, while the invariant-polytope algorithm tries to find a norm that bounds the products of length 1 already enough. The finite tree algorithm, on the other hand, bounds the products using some partitioning-space that separates every product into products that are 1-bounded and some rest-term that doesnt grow fast enough to overcome the k-th root of the JSR definition (polynomial growth). By integrating these algorithms into a hybrid approach, this work aims to advance the computational tools available for JSR analysis combining efficiency and a vast space of matrix sets for which the algorithm terminates.

2 Invariant-polytope algorithm

In this chapter we bring our interest to the underlying invariant-polytope algorithm. One result about JSR computation, that every irreducable family possesses an invariant norm is helpful. We observe that there always exists an norm that is in some sense extremal.

2.1 Extremal norms

Definition 2.1.1. *A norm $\|\cdot\|$ is called invariant if there is a number $\lambda \geq 0$ such that*

$$\max_{j=1,\dots,J} \|A_j x\| = \lambda, \quad \forall x \in \mathbb{R}^d$$

Theorem 2.1.2. *Every irreducible family $\mathcal{A}$ possesses an invariant norm.*

Definition 2.1.3. *A norm $\|\cdot\|$ is called extremal for a family of matrices $\mathcal{A}$ if*

$$\|A_j x\| \leq \mathrm{JSR}(\mathcal{A})\|x\| \forall x \in \mathbb{R}^d \text{ and } A_j \in \mathcal{A}$$

Every invariant norm is extremal

2.2 Invariant polytope norms

For every polytope there exists a corrisponding *Minkowski norm* where the polytope is its unit ball.

2.3 Structure of the invariant-polytope algorithm

After the preprocessing 1.3 is over we start with the leading eigenvector of the calculated candidate as V. Now we add new vertices to V iteratively by multiplying with the matrices in $\mathcal{A}$ and checking if the polytope-norm corrisponding to V of the new vertex is greater than 1. The algorithm terminates when no new vertices are added.

Algorithm 1 invariant-polytope algorithm

V := $\{v_1, \cdots, v_M\}$
$V_{\text{new}} \leftarrow V$
while $V_{\text{new}} \neq \emptyset$ **do**
  $V_{\text{rem}} \leftarrow V_{\text{new}}$
  $V_{\text{new}} \leftarrow \emptyset$
  for $v \in V_{\text{rem}}$ **do**
    for $A \in \mathcal{A}$ **do**
      if $\|Av\|_{\text{co}_s(V)} \geq 1$ **then**
        $V \leftarrow V \cup Av$
        $V_{\text{new}} \leftarrow V_{\text{new}} \cup Av$
return $\text{co}_s(V)$

2.4 Stopping criterions

The runtime of the algorithm 1 is not finite in general. Suitable conditions for stopping or recalculating a candidate have to be put in place. Of course a bare minimum of an max iteration is implemented. The paper (Guglielmi and Protasov, 2013) promises at least good bounds on the real JSR value while also proposing a stopping criterion thats based on eigenplanes. This criterion also generates a new candidate if the last wasnt sufficient after finite time.

2.5 Termination conditions

The invariant-polytope algorithm is very efficient but has its caviat. It only is guaranteed to terminate in finite time if the leading eigenvector of the chosen candidate is unique and simple. In the following i will present the according proofs.

2.6 Rebalancing and added starting vertices

Three years after publishing the fundamental algorithm, the creators released a new paper on rebalancing multiple s.m.ps as well as starting with some extra vertices so the polytope is conditioned better (not as flat).

2.7 Eigenvector cases

If the eigenvector is complex, then a different norm must be used, a so called complex balanced polytope norm. Also in the case of nonnegative matrices a different norm is used.

3 Finite-tree algorithm

In this chapter I want to introduce the finite-tree algorithm and its theoretical background.

3.1 Notation and definitions

Throughout this thesis, we consider a finite set of matrices $\mathcal{A} = \{A_1, \ldots, A_J\}$ with $A_j \in \mathbb{R}^{s \times s}$. The computation of the JSR using tree-based methods, requires a structured representation of matrix products.

Encoding of Matrix Products

The set $\mathcal{I}^n := \{1, \ldots, J\}^n$ denotes the collection of all index sequences of length n. For an index sequence $i = [i_1, \ldots, i_n] \in \mathcal{I}^n$, the corresponding matrix product is given by

$$A_i = A_{i_n} \cdots A_{i_1}.$$

For $n = 0$, we define the identity matrix $A_i = I$.

Definition of an finite-tree

An (A, G)-tree is a structured representation of matrix products used to decompose arbitrary matrix products from $\mathcal{A}$. Lets define it given a set of generator indices $G = \{g_1, \ldots, g_I\} \subseteq J^n$:

Definition 3.1.1. *A Tree with the following structure:*

- *The root node contains the identity matrix: $t_0 = \{I\}$.*
- *Each node $t \in T$ is either:*
  - *A leaf (i.e., it has no children),*
  - *A parent of exactly J children: $\{A_j P : P \in t\}, j = 1, \ldots, J$ (positive children),*
  - *A parent of arbitrarily many generators: $\{A_g^n P : P \in t, n \in \mathbb{N}_0\}$ for some $g \in G$ (negative children).*

Is called $(\mathcal{A}, \mathbf{G})$-tree

Covered Nodes

A node in the tree is called *covered* if it is a subset of one of its ancestors in the tree. Otherwise, it is called *uncovered*.

Definition 3.1.2. *The set of uncovered leaves is denoted as*

$$\mathcal{L}(T) := \{L \in t : t \textit{ is an uncovered leaf of } T\}.$$

and called leafage of the tree T.

1-Boundedness

An (A, G)-tree T is called *1-bounded* with respect to a matrix norm $\|\cdot\|$ if

$$\sup_{L \in \mathcal{L}(T)} \|L\| \leq 1.$$

If a stricter bound holds, i.e., $\sup_{L \in \mathcal{L}(T)} \|L\| < 1$, then the tree is called *strictly 1-bounded*.

3.2 Structure of the finite-tree algorithm

We start with the root node, the identity $t_0 = I$. Now we build up an $(\mathcal{A}, \mathbf{G})$-tree. For that we calculate the norms of the children of current leaf-nodes and if they are bigger than 1 we add them to the tree. The chosen norm in this case is arbitrary but changes the runtime dramatically. If the algorithm terminates a 1-bounded tree was found and by that we have proven that there exists a decomposition for every product of matrices from $\mathcal{A}$.

Algorithm 2 Finite-tree algorithm

Theorem 3.2.1. *If the finite-tree algorithm 2 terminates the* $\mathrm{JSR}(\mathcal{A})$ *was found.*

Proof. The leafage of the generated tree is 1-bounded. Now we take an arbitrary product $P \in \mathcal{A}^n$. If the profduct is outside the current tree i.e. after applying lexicographic ordering on the encodings of the nodes the encoding of the product is bigger then every other encoding in the tree. That means the structure of (A, G)-trees allows us to find one leaf-node that is a valid prefix of the product. Now we have $P = P'L$ with $\|L\| \leq 1$. Since every leaf-node is guaranteed to have at least length 1 (first branches must be $\mathcal{A}$) the length of P' is strictly smaller then the length of P. We can repeat that process until $P'^{(k)}$ is within the tree. If the product is within the tree, we can generate a polynomial $p(k)$ that is monotone and an upper bound on the norms of products within the tree, where k is the length of the product. For that we have to analyse the Jordan-Normalform of the products. Since the spectral radius of every matrix in $\mathcal{A}$ and of every generator is less than 1, we now the growth of potencies of the Jordan-blocks is bounded by a polynomial in the order of the size of the Jordan-block. Just taking a maximum over all factors of all such polynomials we generate a polynomial that bounds every product within the tree according to the length of the product. So every product $P \in \mathcal{A}^n$ is of the form $P = S \cdot P_k \cdot ... \cdot P_1$ where $\|P_i\| \leq 1$ and $\|S\| \leq p(k)$. We imply:

$$JSR(\mathcal{A}) = \lim_k \max_{P \in \mathcal{A}^k} \|P\|^{\frac{1}{k}} \leq \lim_k p(k)^{\frac{1}{k}} = 1$$

The detailed proof can be read in (Möller and Reif, 2014). □

3.3 Efficiency results

Theorem 3.3.1. *If the invariant-polytope algorithm terminates so does the finite-tree algorithm.*

Proof. The proof can be seen in (Möller and Reif, 2014). □

Theorem 3.3.2. *The solution space of the finite-tree algorithm is strictly bigger then the one from the invariant-polytope algorithm.*

Proof. From theorem 3.3.1 we know that the finite-tree algorithm always terminates if the invariant-polytope algorithm does. But there exist cases where the finite-tree algorithm terminates but the ivariant-polytope algorithm does not. □

3.4 Ussage of generators

In (Möller and Reif, 2014), the authors show that the use of generators is essential for ensuring termination in cases where the invariant-polytope algorithm fails to terminate.

4 Hybrid approach

In this chapter we want to explore some possible combinations of the before mentioned algorithm schemes and then present the main result of this work, the tree-flavored-invariant-polytope-algorithm and its termination results.

4.1 Goal and options

In its heart the invariant-polytope algorithm tries to find a polytope, which corrisponding minkowski-norm is specifically optimized on the given problem, whilst the finite-tree algorithm connects growth-rate to decompositions of products. A clear combination scheme arises naturally, where we use the optimized polytope norm to estimate the products of the finite-tree. From there we can choose a specific order or level of concurrency.
The most modular approach would be to first run the invariant-polytope algorithm for a couple of runs and then use the calculated norm thats specially optimized for the finite-tree algorithm. But that seems to be wasteful since valuable matrix calculations from the finite-tree algorithm could have been used for an even more optimized norm and some polytopes might have already cleared insight for the decompositions that the finite-tree algorithm tries to find. In (Mejstrik and Reif, 2024a) the authors came up with a more concurrent algorithm that builds up norms and decompositions in every step.

4.2 Structure of the hybrid algorithm

We try to decompose arbitrary products $P \in \mathcal{A}^k$, such that their polytope-norms are less then $p(k)$ where p is a monotone polynomial. This removes the invariance property of the polytope to be build up, since the norms dont have to be less than 1 but it still proofes the JSR identity because we take the averaged norms in the length of the products in the three-member-inequality 1.3.
Starting the loop of the invariant-polytope algorithm with a cycle on top that is connected via the generators factors and also the first branches represented by images from the missing $J-1$ factors from $\mathcal{A}$. Instead of only adding images under vertices from V and matrices from $\mathcal{A}$ directly, from now on we try to find an $(\mathcal{A}, \mathbf{G})$-tree which is one-bounded i.e its leafage-polytope-norm is less than 1, for every $v \in V$. For that we generate $(\mathcal{A}, \mathbf{G})$-tree patterns in the beginning and just go through every remaining vertex and calculate the leafage-norms. From the structure of those trees we can assume that every matrix in $\mathcal{A}$ represents a node for the first branches. For the branches that lead to a leafage-norm less than or equal 1 we are done, for the other branches we have the choice to go deeper or just add some points to V that changes

the leafage-norm of those branches to less than 1. Here we decided to add the points since going deeper just would mean to consider possibly the same products but the tree generation would be more complex with options for depth-first- or breadth-first-search and even using some s.m.p and generator trickery. [might change it in the future]
First points that come to mind are the leafage-points itself since this is what we have tested but generators could be involved meaning there are possibly infinitely many leafage-points. So the next best thing would be the roots of the branches which are guaranteed to be a single matrix from $\mathcal{A}$. This makes tree generation easy and adds points with likely more distance to the faces of the polytope and makes the norm stronger more quickly.
So in principle for every $v \in V_{\text{rem}}$ take a tree from the generating pool, check the leafage-norm for every root branch, if it is larger than 1 add the point from the root branch to V_{new} and V. Repeat this as long as new vertices have been added. We use V for the polytope-norms and since new points are only being added the norms decrease over time so all 1-bounded trees stay bounded.
After termination the set of trees generated promise a valid decomposition for every product from $\mathcal{A}$ into chunks of norm lesser 1 and one suffix thats of norm less than $p(k)$ for some monotone polynomial, which proofes the question if the chosen radius is maximal.

Algorithm 3 Tree-flavored-invariant-polytope-algorithm

$V := \{v_1, \cdots, v_M\}$
$V_{\text{new}} \leftarrow V$
while $V_{\text{new}} \neq \emptyset$ **do**
    $V_{\text{rem}} \leftarrow V_{\text{new}}$
    $V_{\text{new}} \leftarrow \emptyset$
    for $v \in V_{\text{rem}}$ **do**
        Construct some $(\mathcal{A}, \mathbf{G})$-tree $\mathbf{T}$
        for $L = L'A \in \mathcal{L}(T)$ with $A \in \mathcal{A}$ **do**
            if $\|Lv\|_{\text{co}_s(V)} \geq 1$ **then**
                VAv
                $V_{\text{new}} \leftarrow V_{\text{new}} \cup Av$
return

The following figure 4.1 shows an example of the generated tree structure as well as the tested $(\mathcal{A}, \mathbf{G})$-trees that have been used to bound the products. Here $\mathcal{A}$ consists of two matrices A and B and the chosen candidate is $\Pi = ABA$. At the top you can see the cycle generated by the candidate and the starting vector v_1 as well as branches coming of, that are the products that stay in contrast to the cycle. This is what i call the crown and it is generated for every matrix set at the beginning, so no testing has been made so far. All the nodes that are connected through a solid arrow have been added to the vertices V.

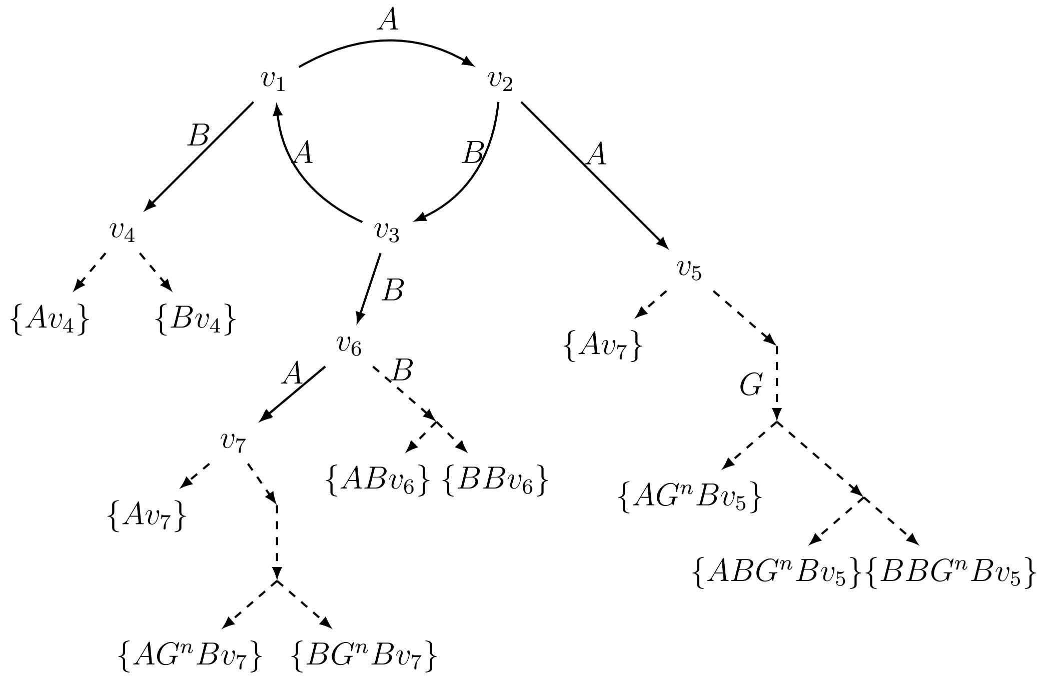

Figure 4.1: Cyclic tree structure generated by the algorithm. Vertices added to V (solid arrows) and finite-trees for bounding products (dashed arrows).

Now the finite-tree theory comes in and every vertex gets tested by a $(\mathcal{A}, \mathbf{G})$-tree (dashed arrows). Take the vector v_6 for example it has been tested by an tree and the branch starting with the matrix B was sufficient i.e. every leaf-node has a polytope-norm of less than 1 (at the time the polytope consists of the vertices $V = \{v_1, \cdots, v_6\}$). The branch starting with the matrix A on the other hand did not, so the vertex $v_7 = Av_6$ has been added to V and the next tree was tested, which was fully sufficient (on every branch). Since the other branches also terminated earlier the algorithm stops, no more vertices are beeing added and the candidate Π was proofen to be a s.m.p product.

4.3 Termination results

Lemma 4.3.1. *For every cyclic-tree T generated by algortihm 3, there exists a monotone polynomial p that bounds every product P encoded by nodes of the tree in the produced polytope-norm and according to the length $|P|$ of the product (number of used factors from $\mathcal{A}$):*

$$\exists p : \forall P \in T : \|P\|_V \le p(|P|)$$

Proof. Take an arbitrary node $N = [i_1, \cdots, i_n]$ of the generated tree. Now every i_j encodes either a matrix from $\mathcal{A}$ or a generator from $\mathcal{G}$. Since all those matrices have a spectral radius less then 1, its Jordan-Normal forms grow utmost polynomially und thus, due to equality of norms in finite-dimensional vector spaces, we have that $\exists p_{i_j} : \|A_{i_j}\|_V \le p_{i_j}(|A_{i_j}|)$, where the number of factors only matters for generators since they are encoded for any exponent $n \in \mathbb{N}$.

Here A_{i_j} stand for the actual factors from an arbitrary product encoded by N so that $|A_{i_j}|$ is well defined. Now we take the product of those polynomials and get an upper bound for every product encoded by this node.

$$\text{Let:} \quad k = \sum_{j=1}^{n} |A_{i_j}| \quad \text{and} \quad p_N = \prod_{j=1}^{n} p_{i_j}$$

$$\begin{aligned} \|A_{i_n} \cdot \ldots \cdot A_{i_1}\|_V \leq \|A_{i_n}\|_V \cdot \ldots \cdot \|A_{i_1}\|_V &\leq p_{i_n}(|A_{i_n}|) \cdot \ldots \cdot p_{i_1}(|A_{i_1}|) \\ &\leq p_{i_n}(k) \cdot \ldots \cdot p_{i_1}(k) = p_N(k) \end{aligned}$$

So since the node was arbitrary we have such a polynomial p_N for every node $N \in T$.

$$\text{Now let} \quad p_N = \sum_{r \geq 0} \lambda_{N,r} x^r \quad \text{then} \quad p = \sum_{r \geq 0} \max_{N \in T} \lambda_{N,r} x^r$$

and due to the fact that the tree has only finitely many nodes, the maximum is actually well defined. This concludes the lemma.

□

Lemma 4.3.2. *There exists an monotone polynomial p such that for every product $P \in \mathcal{A}^k$ and $v \in V$, $\|Pv\|_V \leq p(k)$ holds.*

Proof. Since for every $v \in V$ there exists an $(\mathcal{A}, \mathbf{G})$-tree that is 1-bounded, we can find a leaf L and according suffix P' such that $P = P'L$ and $Lv = \sum_{i=1}^{J} \lambda_i v_i$ with $\sum_{i=1}^{J} |\lambda_i| \leq 1$.
Now we have:

$$\|Pv\|_V = \|P'Lv\|_V \; = \; \|P' \sum_{i=1}^{J} \lambda_i v_i\|_V \leq \sum_{i=1}^{J} |\lambda_i| \|P'v_i\|_V$$

This means we got to the same point as the beginnig in bounding the product of an arbitrary P' and $v \in V$. Since the leafs of an $(\mathcal{A}, \mathbf{G})$-tree have always one or more factors, the sice of P' decreased from the sice of P. So if we repeat the process enough times P' lies within the tree T and therefore, according to lemma 4.3.1, there exists a monotone polynomial p such that:

$$\|P'\|_V \leq p(|P'|) \leq p(|P|)$$

This implies:

$$\|Pv\|_V \leq \sum_{i=1}^{J} |\lambda_i| \|P'v_i\|_V \leq \sum_{i=1}^{J} |\lambda_i| \|P'\|_V \leq \sum_{i=1}^{J} |\lambda_i| p(|P|) \leq p(|P|)$$

Which concludes the proof. □

Theorem 4.3.3. *If Algorithm 3 terminates then $JSR(\mathcal{A}) \leq 1$*

Proof. Due to the prework in the lemmas this is now easy to show. Lemma 4.3.2 implies that for every $P \in \mathcal{A}^k$ and $v \in V$, $\|Pv\|_V \leq p(k)$ trivially this implies:

$$\forall P \in \mathcal{A}^k : \|P\|_V \leq p(k)$$

Now we take the definition of the JSR and get:

$$JSR(\mathcal{A}) = \lim_{k \geq 1} \max_{P \in \mathcal{A}^k} \|P\|^{\frac{1}{k}} \leq \lim_{k \geq 1} p(k)^{\frac{1}{k}} = 1$$

□

Of course, if the candidate had a spectral radius of $1 \implies JSR(\mathcal{A}) = 1$.

Corollary 4.3.4. *If the chosen trees are always $T_v = \mathcal{A}$, then we would have exactly the invariant-polytope algorithm 1. Which makes the new approach a real generalization.*
Also the polynomial bounding would collapse to just a constant $K = \max_{v \in V} \max_{P \in T_v} : \|P\|_{co_s(V)}$ that is the maximum over all polytope-norm values of nodes in the generated cyclic-tree. This constant would not depend on the sizes of the products anymore.

4.4 Efficiency

Remarks

Used trees

It was proofen in (Möller and Reif, 2014) that in order to have a bigger solution space then algorithm 1 the use of generators is mandatory. The choice of the testing-trees is not trivial and subject to active research. We propose to use so-called minimal trees that fulfill every necessary and beneficial condition of the $(\mathcal{A}, \mathbf{G})$-trees but keep the amount of nodes minimal.

Stopping criterions

Some of the stopping criterions of the invariant-polytope algorithm can still be used as the structure is very similar. For this the eigenplane stopping condition is implemented.

Polytope invariance

Since the polytope-norms of matrices in $\mathcal{A}$ are allowed to be bigger than 1 just less than $p(1)$ it is possible and very likely the polytope is not invariant.

4.5 Implementation

Definition 4.5.1. *An $(\mathcal{A}, \mathbf{G})$-tree with just one generator at the root, one path from root to the covered node and all neccessarily added branches, to make it comply with definition 3.1, is called* *a* minimal tree. *This tree only depends on the set* $\mathcal{A}$ *and the chosen generator* G. *Its denoted by* $\min(G, \mathcal{A})$.

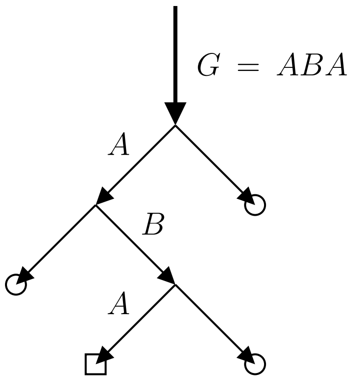

Figure 4.2: Minimal tree structure $\min(G, \mathcal{A})$ for the candidate $G = ABA$ and $\mathcal{A} = \{A, B\}$. The arrows marked with circles are uncovered-leafs, the arrow marked with a square is a covered-leaf

5 Numerics

5.1 Runtime of algorithm 3 on generated matrices

5.1.1 Variing matrix dimensions

5.1.2 Variing matrix condition

5.1.3 Variing matrix s.m.p number

5.2 Comparison algorithm 3 and 1

5.2.1 Big matrix dimensions

5.2.2 Multiple s.m.p's

5.2.3 old closed problems couldnt be solved by 1

5.3 Comparison algorithm 3 and 2

5.3.1 Multiple s.m.p's and big matrix dimensions

5.3.2 old closed problems couldnt be solved by 2

5.4 Solving open problems

6 Conclusion

6.1 New approach

6.2 Solved Problems

6.3 Efficiency

6.4 Unsolved problems

6.5 Further research

6.5.1 Complex case

6.5.2 Nonnegative case

6.5.3 Optimizing tree generation and vertex choice

Bibliography

Blondel, V. D. and Tsitsiklis, J. N. (2000). A survey of computational complexity results in systems and control. *Automatica*, 36(9):1249–1274.

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

Guglielmi, N. and Protasov, V. (2011). Exact computation of joint spectral characteristics of linear operators.

Guglielmi, N. and Protasov, V. (2013). Exact Computation of Joint Spectral Characteristics of Linear Operators. *Foundations of Computational Mathematics*, 13(1):37–97.

Guglielmi, N. and Protasov, V. Y. (2015a). Invariant polytopes of linear operators with applications to regularity of wavelets and of subdivisions. *none*.

Guglielmi, N. and Protasov, V. Y. (2015b). Invariant polytopes of linear operators with applications to regularity of wavelets and of subdivisions.

Heil, C. (1993). Ten Lectures on Wavelets (Ingrid Daubechies). *SIAM Review*, 35(4):666–669.

Jungers, R. M. (2009). The Joint Spectral Radius. *none*.

Mejstrik, T. (2020a). Algorithm 1011: Improved Invariant Polytope Algorithm and Applications. *ACM Transactions on Mathematical Software*, 46(3):1–26.

Mejstrik, T. (2020b). Improved invariant polytope algorithm and applications.

Mejstrik, T. and Reif, U. (2024a). A Hybrid Approach to Joint Spectral Radius Computation. *none*.

Mejstrik, T. and Reif, U. (2024b). Hybrid approach to the joint spectral radius computation.

Möller, C. and Reif, U. (2014). A tree-based approach to joint spectral radius determination. *Linear Algebra and its Applications*, 463:154–170.

Rota, G.-C. and Gilbert Strang, W. (1960). A note on the joint spectral radius. *Indagationes Mathematicae (Proceedings)*, 63:379–381.