NONLINEAR EIGENVALUE PROBLEMS: THEORY AND ALGORITHMS

A thesis submitted to The University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Science and Engineering

2021

Gian Maria Negri Porzio
Department of Mathematics

CONTENTS

Word count: 70268

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

In this thesis we focus on the theoretical and computational aspects of nonlinear eigenvalue problems (NEPs), which arise in several fields of computational science and engineering, such as fluid dynamics, optics, and structural engineering. In the last twenty years several researchers devoted their time in studying efficient and precise ways to solve NEPs, which cemented their importance in numerical linear algebra. The most successful algorithms developed towards this goal are either based on contour integrals, or on rational approximations and linearizations.

The first part of the thesis is dedicated to contour integral algorithms. In this framework, one computes specific integrals of a holomorphic function $G(z)$ over a contour $\partial\Omega$ and exploits results of complex analysis to retrieve the eigenvalues of $G(z)$ inside Ω. Our main contribution consists in having expanded the theory to include meromorphic functions, i.e., functions with poles inside the target region Ω. We showed that under some circumstances, these algorithms can mistake a pole for an eigenvalue, but these situations are easily recognised and the main results from the holomorphic case can be extended. Furthermore, we proposed several heuristics to automatically choose the parameters that are needed to precisely retrieve the eigenpairs.

In the second part of the thesis, we focus on rational approximations. Our goal was developing algorithms that construct robust, i.e., reliable for a given tolerance and scaling independent, rational approximations for functions given in split form or in black-box form. In the first case, we proposed a variant of the set-valued AAA, named weighted AAA, which guarantees to return an approximation with the chosen accuracy. In the second one, we built a two-phase approach, where an initial step performed by the surrogate AAA is followed by a cyclic Leja–Bagby refinement. We concluded the section with numerous numerical experiments based on the NLEVP library, comparing contour integral and rational approximation algorithms.

The third and final part of the thesis is about tropical linear algebra. Our research on this topic started has a way to set the parameters of the aforementioned contour integral algorithms: in order to do that, we extended the theory of tropical roots from tropical polynomials to tropical Laurent series. Unlike in the polynomial case, a tropical Laurent series may have infinite tropical roots, but they are still in bijection with the slopes of the associated Newton polygon and they still provide annuli of exclusion for the eigenvalues of the Laurent series.

DECLARATION

No portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

COPYRIGHT STATEMENT

ACKNOWLEDGEMENTS

This thesis marks the end of my PhD, hence is the perfect place to ponder on my three years and half spent at the University of Manchester. First, I would like to express my deepest gratitude to my supervisor, Prof. Françoise Tisseur. She has always been supportive and ready to give precious advice. I could not have wished for a better mentor, both humanly and professionally. Further, I would like to thank all the other members of the NLA Group, especially Prof. Nicholas J. Higham and Dr. Stefan Güttel, whom I had the pleasure to work with and have been continuous sources of inspirations. I also thank my examiners, Dr. Marcus Webb and Prof. Marc Van Barel, who helped me improve this work with their advice.

I thank my parents, Oriana and Luigi, who have always been there during all this time, from when I was a child to the entirety of my academic career. If I became the man I am today, I owe it all to them. Similarly, I cannot forget my extended family, so I am grateful to Gianfilippo, Tiziana, Gianluca and Luana for the precious time we spent together, even if it was not a lot given the distances across us.

The pandemic was a hard toll on everyone of us and we will remember it for our lifetime, from when we thought it would end in a couple of months to when we got used to work and teach from home. Writing the thesis far away from the Department and without the everyday routine added another layer of difficulty. Nevertheless, I consider myself lucky because I finally had the possibility to live with my girlfriend, Lucia. Thank you, my dear: without you this period would have been unbearable.

I shared Office 2.111 with fantastic people. Thank you Mante, Mila, Tom, Michael, Bob, and Xynie for all the nice time we spent on solving problems – and all the other moments I should not mention here. A special thank goes to Massimiliano, with whom I shared many late night sessions, where we thought about our research between Korean take-away or a home-made dinner.

My last acknowledgements go to my other colleagues. Marco, thanks for your support, the experiences and the trips together; Natasha and Gab, you have been wonderful flatmates; Dom, Floriana, and Filippo, thank you for all the sports and the dinners we organised. Finally, my apologies go to all the people that left a mark on my life in Manchester, but I could not include in these pages. Even if your name does not appear, I surely recall with joy every moment we spent together. If you ever read these pages, I wish all of you a bright and happy future.

PUBLICATIONS

- Chapter 2 is partially based on the manuscript: Gian Maria Negri Porzio and Françoise Tisseur. *Contour integral methods for meromorphic eigenvalue problems*, currently under preparation.
- Chapter 3 is partially based on the manuscript above-mentioned and on the technical report: Nicholas J. Higham, Gian Maria Negri Porzio, and Françoise Tisseur. An Updated Set of Nonlinear Eigenvalue Problems. MIMS EPrint 2019.5, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, 2019.
- Chapter 4 is based on the article: Stefan Güttel, Gian Maria Negri Porzio, and Françoise Tisseur. Robust Rational approximations of Nonlinear Eigenvalue Problems. Submitted to *SIAM J. Sci. Comput.*, currently under revision.
- Chapter 5 is based on the manuscript: Gian Maria Negri Porzio, Vanni Noferini, and Leonardo Robol. Tropical Laurent series, their tropical roots, and localization results for the eigenvalues of nonlinear matrix functions. Submitted to Mathematics of Computation.

1 INTRODUCTION

Often during my experience as a PhD student, family and friends asked what I was studying and what I was working on. At the beginning, my answers were a mix of babbling and hesitation: how could I have explained nonlinear eigenvalue problems to them? Nevertheless, after a while I became an expert of this art and, in hindsight, the solution was natural: eigenvalue problems are everywhere around us and bringing examples is the best way to explain what we study to non-technical acquaintances. For instance, the PageRank algorithm transformed the way people surf on the web; quadratic eigenvalue problems lie behind most of the model of mechanical systems, such as vibration analysis and fluid dynamics; exponential problems are used to model delay-differential equations (DDE), for example, in electronic devices. In the last twenty years researchers have made great strides in linear, but also polynomial and rational eigenvalue problems. Therefore it is in human nature to tackle generalisations and asking further questions.

The goal of this thesis is adding another piece to the puzzle that are the nonlinear eigenvalue problems, with the hope of enlightening the next small section of the knowledge path to future researchers. This chapter serves as a smooth transition to the technical parts of the thesis. It is structured as follows. In Section 1.1 we revise the mathematical background the reader should master before approaching the core of the thesis. In Section 1.2 we describe the nonlinear eigenvalue problem, mainly pointing out the differences with its simpler linear counterpart. Finally, Section 1.3 is dedicated to the descriptions of real world applications, while in Section 1.4 we delineate the structure of the thesis.

1.1 PREREQUISITES

The main goal of this section is setting the necessary definitions and notation that we will be using throughout the thesis. We start with the basics: given a matrix $A \in \mathbb{C}^{n \times n}$, we say that $\lambda \in \mathbb{C}$ is an *eigenvalue* of A if

$$\det(A - \lambda I) = 0,$$

or, equivalently, if there exists a *right eigenvector* and a *left eigenvector* $v, w \in \mathbb{C}^{n \times n} \backslash \{0_n\}$ such that

$$\begin{aligned} (A - \lambda I)v &= 0_n, \\ w^*(A - \lambda I) &= 0_n^*. \end{aligned} \tag{1.1}$$

We will usually drop the adjective "right" and we will say the "eigenvector v" [GV96, Chapter 2]. This does not cause any loss of generality, because the left eigenvectors w of A are the right eigenvectors of A^*. The set

$$\Lambda(A) := \{\lambda : \det(A - \lambda I) = 0\}$$

is called the *spectrum* of A and consists of all the eigenvalues of A [GV96, Chapter 7]. It is evident that the number of distinct eigenvalues, say s, is always less or equal than n. Further, the multiplicity of λ as a root of $\det(A - \lambda I)$ is called the *algebraic multiplicity* of λ. Similarly, the eigenvectors of λ, together with the zero vector, form a subspace of $\mathbb{C}^{n \times n}$, which is called the *eigenspace* of λ. The dimension of the eigenspace is called the *geometric multiplicity* of λ and one can prove that it is always less or equal than its algebraic multiplicity. We say that an eigenvalue is *simple* when the algebraic multiplicity is equal to 1, and it is *semisimple* when the geometric multiplicity is equal to the algebraic one. When it is strictly less, we can find the so-called *generalised eigenvectors*; the subspace spanned by the eigenvectors and all the generalised eigenvectors of λ is equal to the algebraic multiplicity. In addition, (generalised) eigenvectors of different eigenvalues are independent from each other. Hence, all together they form a basis of $\mathbb{C}^n$ and allow us to decompose A in its *Jordan Canonical Form*.

Definition 1.1 (Jordan Canonical Form [GV96, Chapter 7])**.** Consider $A \in \mathbb{C}^{n\times n}$. Then there exists an invertible matrix U such that

$$A = U^{-1}JU = \operatorname{diag}(J_1, \dots, J_p)$$

where J_k is a *Jordan block* and has the form

$$J_k = \begin{bmatrix} \lambda_k & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_k \end{bmatrix}.$$

Further, J is unique up to block permutations.

Now we focus our attention on a target complex region $\Omega \subset \mathbb{C}$. If not specified otherwise, Ω is a nonempty, open, simply connected, bounded set. Furthermore, we assume its contour $\partial\Omega$ is piecewise C^1, so that we can compute integrals over it. In addition, we assume $\Omega_0 \supset \Omega$ is an open neighbourhood of Ω. The canonical example of Ω is an open disk centered in γ with radius $\rho > 0$, which we denote with $\mathcal{D}(\gamma, \rho)$. Matrix-valued functions are the second main ingredient. We write

$$\begin{aligned} G : \Omega_0 &\rightarrow \mathbb{C}^{n\times n}. \\ z &\mapsto G(z) = \textstyle\sum_{j=1}^{s} g_j(z) A_j, \end{aligned} \tag{1.2}$$

and we say that $G(z)$ is in *split form* if we have direct access to the scalar functions $g_j(z)$ and to the matrix coefficients A_j. Alternatively, we say that $G(z)$ is in *black-box form* if we are only allowed to compute its values at specific points z_i and we have no information on the scalar functions $g_j(z)$ or the matrix coefficients A_j.

We mainly work with a specific class of smooth functions.

Definition 1.2 (Holomorphic functions [Car95])**.** Let $G\colon \Omega_0 \to \mathbb{C}^{n\times n}$ be a matrix-valued function. If the derivative of $G(z)$ at the point z_0

$$G'(z_0) = \lim_{z\to z_0} \frac{G(z) - G(z_0)}{z - z_0}$$

is well-defined at every $z_0 \in \Omega_0$, we say that $G(z)$ is a *holomorphic function (from Ω_0 to $\mathbb{C}^{n\times n}$)* and we write $G \in \mathcal{H}(\Omega_0, \mathbb{C}^{n\times n})$ or $G \in \mathcal{H}(\Omega_0)$ when there are no ambiguities.

Simple examples of holomorphic functions are the polynomials, the exponential, the sine, and the cosine. Rational functions are generally not holomorphic, because their derivative is not defined at the points where they are equal to infinity. More precisely, given $G(z)$ as in (1.2), we say that ξ is a *pole* of $G(z)$ if there exists j such that $1/g_j(\xi) = 0$. A function with this property is said to be *meromorphic*.

Definition 1.3 (Meromorphic functions [Car95])**.** Consider $F\colon \Omega_0 \to \mathbb{C}^{n\times n}$. If $F(z)$ is holomorphic on Ω_0 except on a set of isolated poles, then we say that $F(z)$ is a *meromorphic function (from Ω_0 to $\mathbb{C}^{n\times n}$)* and we write $F(z) \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$ or alternatively $F(z) \in \mathcal{M}(\Omega_0)$ when there are no ambiguities.

From now on, we will use the letter G when we want to underline that the function is holomorphic, while the letter F will be suited for the meromorphic case. Let $\xi \in \Omega_0$ be a pole of $F(z)$. The *multiplicity* of ξ is defined as the smallest integer $c \geqslant 1$ such that $(z - \xi)^c F(z)$ is holomorphic at ξ; if c is equal to one, we say that ξ is *simple*.

Notice that we can always write a meromorphic function $f(z)$ as $f(z) = g_1(z)/g_2(z)$, with $g_i(z) \in \mathcal{H}(\Omega_0, \mathbb{C})$. Further, assume $F(z) \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$: given that the set of poles of $F(z)$ is always discrete, then it is compact in $\overline{\Omega} := \Omega \cup \partial\Omega$, hence there always exists a polynomial $g(z) \in \mathbb{C}[z]$ whose roots are the poles of $F(z)$ in Ω such that $G(z) := g(z)F(z) \in \mathcal{H}(\Omega, \mathbb{C}^{n\times n})$.

These two classes of functions have several properties. For example, all the derivatives $G^{(n)}(z)$ exist, even though Definition 1.2 requires only the first one to be defined. This allows us to define $g(A)$ thanks to the Jordan canonical form.

Definition 1.4 (Matrix function via Jordan Canonical Form [Hig08, Chapter 1]). Let $g(z) \in \mathcal{H}(\Omega_0, \mathbb{C})$, let A have the Jordan canonical form $A = U^{-1}JU$, and assume that $\Lambda(A) \subset \Omega$. Then

$$g(A) := U^{-1}g(J)U,$$

where

$$g(J_{ij}) = \begin{bmatrix} g(\lambda_i) & g'(\lambda_i) & \dots & \frac{g^{(m_{i,j}-1)}(\lambda_i)}{(m_{i,j}-1)!} \\ & g(\lambda_i) & \ddots & \vdots \\ & & \ddots & g'(\lambda_i) \\ & & & g(\lambda_i) \end{bmatrix} \in \mathbb{C}^{m_{i,j} \times m_{i,j}}$$

It is possible to define $g(A)$ in another way, but first we need to state two important results by Cauchy.

Theorem 1.1 (Cauchy's Integral Theorem [AF21]). *Let $g(z) \in \mathcal{H}(\Omega_0, \mathbb{C})$. Then*

$$\int_{\partial\Omega} g(z)\, dz = 0.$$

Theorem 1.2 (Cauchy's Integral Formula [AF21]). *Let $g(z) \in \mathcal{H}(\Omega_0, \mathbb{C})$, $a \in \Omega$. Then*

$$g(a) = \frac{1}{2\pi\mathrm{i}} \int_{\partial\Omega} \frac{g(z)}{z-a}\, dz.$$

Cauchy's integral formula naturally extends to matrices $A \in \mathbb{C}^{n\times n}$ and hence we have the following elegant statement of $g(A)$ for $g \in \mathcal{H}(\Omega_0, \mathbb{C})$, which is equivalent to Definition 1.4.

Theorem 1.3 (Matrix function via Cauchy's Integral Formula [Hig08, Chapter 1]). *Let $g(z) \in \mathcal{H}(\Omega_0, \mathbb{C})$ and let $A \in \mathbb{C}^{n\times n}$ such that $\Lambda(A) \subset \Omega$. Then*

$$g(A) := \frac{1}{2\pi\mathrm{i}} \int_{\partial\Omega} g(z)(zI - A)^{-1}\, dz.$$

Theorem 1.4 (Identity theorem [AF21]). *Let $F_1(z), F_2(z) \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$ and let $(z_j)_{j\in\mathbb{N}} \subset \Omega$ be a sequence with an accumulation point in Ω. If $F_1(z_j) = F_2(z_j)$ for every $j \in \mathbb{N}$, then $F_1(z) \equiv F_2(z)$ in Ω.*

The previous results underline how smooth holomorphic functions are. Cauchy's integral formula tells us that as soon as we know their values on $\partial\Omega$, then we know their values in all Ω. Similarly, the Identity theorem reveals that we only have to know the values of $F(z)$ on a sequence with an accumulation point to identify it. This also implies that all the zeros of a nonzero meromorphic function are discrete, just like their poles. Finally, we can say a lot more on the number of zeros (and poles) of $f(z) \in \mathcal{M}(\Omega_0, \mathbb{C})$ in Ω: it only depends on the values of $f(z)$ and $f'(z)$ on $\partial\Omega$.

Theorem 1.5 (Cauchy's argument principle [AF21]). *Let $f(z) \in \mathcal{M}(\Omega_0, \mathbb{C})$ and assume it has n_r zeros and n_p poles in Ω and no poles nor zeros on $\partial\Omega$. Then*

$$n_r - n_p = \frac{1}{2\pi \mathrm{i}} \int_{\partial\Omega} \frac{f'(z)}{f(z)}\, dz.$$

1.2 THE NONLINEAR EIGENVALUE PROBLEM

Definition 1.5 (Nonlinear eigenvalue problem). Consider $F(z) \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$. The *nonlinear eigenvalue problem* (NEP) consists in finding all the scalars $\lambda \in \Omega$ where $F(z)$ is well-defined, the *eigenvalues*, and the corresponding nonzero vectors $v, w \in \mathbb{C}^n \backslash \{0\}$, the *right and left eigenvectors*, such that

$$F(\lambda)v = 0, \qquad w^* F(\lambda) = 0. \tag{1.3}$$

The pair (λ, v) is a *right eigenpair* of F, while (λ, w^*) is a left eigenpair.

Remark 1.1. Asking that $F(z)$ is well-defined at the eigenvalue λ is a technicality to avoid the edge-case scenario where λ is a pole, but also satisfies (1.3). For example, without that hypothesis $\lambda = 0$ would be an "eigenvalue" of

$$F(z) = \begin{bmatrix} z & z^{-1} \\ 0 & z^{-1} \end{bmatrix}$$

with e_1 as an eigenvector. In addition, (1.3) would not be equivalent to $\det(F(\lambda)) = 0$.

We can easily see that (1.3) is a generalisation of (1.1), where $F(z) = A - zI$. Hence, most of the terminology for the nonlinear eigenvalue problem comes directly from its linear counterpart. For instance, the algebraic and the geometric multiplicity of an eigenvalue λ are defined in the same way, and the same is true for simple and semisimple eigenvalues. Finally, we denote the spectrum of $F(z)$ by $\Lambda(F)$, the eigenspace of λ by $\operatorname{null}(F(\lambda))$, and we say that $F(z)$ is *regular* if $\det(F(z))$ does not vanish identically in Ω.

Remark 1.2. The regularity condition implies that the set of eigenvalues of $F(z)$ is finite in Ω. In fact, if it were not finite, then it would have an accumulation point in $\overline{\Omega}$, which would yield that $\det(F(z)) \equiv 0$ in Ω by the Identity theorem 1.4.

There have been intermediate steps between $(A - \lambda I)v = 0$ and $F(\lambda)v = 0$. As mentioned, everything started with the

- *linear eigenvalue problem* (LEP)

  $$(A - \lambda I)v = 0,$$

  where $A \in \mathbb{C}^{n\times n}$. One of the most well-studied problems in linear algebra, nowadays the golden standard to solve it for small, dense, and unstructured matrices is the QR algorithm [Fra61a; Fra61b; Kub61], while the Arnoldi algorithm with its generalisations is best suited for large and sparse matrices, where only a small subset of the eigenpairs is needed. When the identity matrix is substituted by any matrix $B \in \mathbb{C}^{n\times n}$, then we have the *generalised eigenvalue problem*.

Similar procedures can be used for the GEP, such as the QZ algorithm by Moler and Stewart [MS73].

- The next step is the *quadratic eigenvalue problem* (QEP)

  $$(\lambda^2 M + \lambda C + K)v = 0,$$

  where $M, C, K \in \mathbb{C}^{n\times n}$. Quadratic eigenvalue problems usually appear in second-order differential equations, fluid dynamics, and vibration systems, hence the M, C, K are the ordinary letters used in these problems, because they represent the mass term, the damping term, and the stiffness term matrix, respectively. We direct the reader to the monumental survey by Tisseur and Meerbergen and its references for the properties of the QEP [TM01]. In 2013 Hammarling, Munro, and Tisseur introduced the QUADEIG algorithm, which is backward stable for problems which are not too heavily damped [HMT13]. In 2014, Zeng and Su used a tropical scaling in conjunction with a modification of QUADEIG to develop an algorithm which is always backward stable [ZS14].

- Quadratic eigenvalue problems are a subclass of the *polynomial eigenvalue problem* (PEP)

  $$\left(\sum_{j=0}^{d} \lambda^j A_j\right) v = 0,$$

  where $A_j \in \mathbb{C}^{n\times n}$. Similarly to the QEPs, they often appear in fluid dynamics [Ors71], optics [ZL08], and plasma physics [TKL05]. A standard algorithm to solve them is *linearization*, where the original $n \times n$ problem becomes a larger linear one with the same spectral structure. We refer to [ACL09; Cha+19] and [Mac+06b; Mac+06a] for further details.

- *Rational eigenvalue problems* (REP) have the form

  $$\left(\sum_{j=1}^{k} \frac{n_j(\lambda)}{d_j(\lambda)} A_j\right) v = 0,$$

where $A_j \in \mathbb{C}^{n\times n}$ and $n_j(z), d_j(z)$ are scalar polynomials. We refer to [SB11] for applications and a general overview of this type of problems.

The term "nonlinear" in Definition 1.5 is somewhat ambiguous. First, Equation (1.3) is indeed nonlinear in the eigenvalue λ, but is linear in the eigenvector v. Problems where the nonlinearity appears in the eigenvector as well go beyond the scope of this thesis. Furthermore, some works use this term for matrix-valued functions that are, in fact, "nonlinear", hence polynomials and rational functions are included; others, this thesis included, focus on problems where linearization techniques cannot be directly applied, thus excluding polynomials and rational functions. Probably "holomorphic eigenvalue problems" (HEP) or "meromorphic eigenvalue problems" (MEP) would be better, non-ambiguous choices: unfortunately, they never became established in the numerical linear algebra community. Finally, we remark that the abbreviation NEP is not universal: in the literature one may find NLEP, NLEVP, or IEVP (interior eigenvalue problem [Mor20]). In this thesis we chose NEP for the symmetry with the other classes of eigenvalue problems.

1.2.1 Eigenvalues and eigenvectors

Another way to see (1.3) is as a generalisation of the *root finding problem* in higher dimensions: given a sufficiently smooth function $f\colon \mathbb{C} \to \mathbb{C}$, find λ such that $f(\lambda) = 0$. This point of view shows one of the major differences when focusing on NEPs: while for the other classes of eigenvalue problems the number of eigenvalues is a function of the size n (e.g., a matrix polynomial of degree d has always dn eigenvalues in $\mathbb{C} \cup \{\infty\}$), this is no longer true in the nonlinear case. In fact, a regular nonlinear problem may have

- zero eigenvalues, e.g., $f(z) = e^z$;
- finitely many eigenvalues, e.g., $f(z) = e^z - z^4 + 2$;
- countably infinite eigenvalues, e.g., $f(z) = \cos(z)$.

Concerning the algebraic and geometric multiplicity of the eigenvalues, as in the linear case the latter is smaller than the former. However, it is no longer true they are bounded by the dimension n, see, for instance, $f(z) = z^d$. Furthermore, the (generalised) eigenvectors of different eigenvalues are neither independent from each other (e.g., $f(z) = z(z-1)$). We will focus more on these aspects at the beginning of Chapter 2.

Example 1.1. *Consider the matrix-valued function*

$$F(z) = \begin{bmatrix} e^{iz^2} & 1 \\ 1 & 1 \end{bmatrix}, \tag{1.4}$$

which first appeared in [GT17] and is available in the NLEVP *library since version 4.0 as* `nep1` *[Bet+11]. Its spectrum consists of the points $\{\pm\sqrt{2\pi k} \mid k \in \mathbb{Z}\}$, with $e_1 - e_2$ being an eigenvector for all of them. All the eigenvalues except for $\lambda_0 = 0$ are simple, while the algebraic multiplicity of λ_0 is* 2, *and the geometric multiplicity is* 1.

1.2.2 Classes of algorithms

Even though it does not exist yet a clear best solver for nonlinear eigenvalue problems, each algorithm broadly falls in one of the following categories: solvers based on Newton's method, solvers based on contour integrals, and solvers based on linearisations. In this section we briefly review them and recommend further readings or previous works on these topics.

Algorithms based on the Newton-Raphson method

The Newton-Raphson's (or simply Newton's) method is a natural way to find eigenvalues and eigenvectors of nonlinear functions. In its simplest form, it is an iterative method to find the simple roots of a scalar function $f(z)$ given the initial point z_0. The iteration reads

$$z_{k+1} = z_k - \frac{f(z_k)}{f'(z_k)}.$$

If the initial guess z_0 is close enough and the function is at least C^1, then the convergence is quadratic. Hence, it seems a valid algorithm to find nonlinear eigenpairs.

In turn, Newton's methods can be either applied on a scalar function, such as $\det F(z)$, in order to find only the eigenvalues; or to the original eigenproblem itself to extract both the eigenvalues and the eigenvectors. In the former category we have the Newton-trace iteration by Lancaster [Lan02], the Newton-QR iteration by Kublanovskaya [Kub70], or the BDS (border, deletion, substitution) method by Andrew, Chu, and Lancaster [ACL95]. In the latter we recall the nonlinear inverse iteration, which was mentioned in 1950 by Unger [Ung50] and deeply analysed by Ruhe in 1973 [Ruh73].

Among the three categories of nonlinear eigensolvers, the ones based on Newton's method present the most serious disadvantages. Even though they only require one parameter to start, i.e., the initial guess z_0, it is fundamentally important: if it is too far away from the target eigenvalue, then we may not converge there. In addition, they compute only a single eigenvalue each time we complete a set of iterations, therefore it may become painfully slow if our goal is computing a large chunk of the spectrum. In Section 3.5 of Chapter 3 we will discuss some of these methods with further details.

Algorithms based on linearizations

Since many years linearisation techniques allow to reformulate any polynomial eigenvalue problem as a larger linear eigenvalue problem [ACL09; Mac+06b; Mac+06a]. In 2011, Su and Bai proposed a linearization for rational eigenvalue problems which preserved the low-rank structure of the matrix coefficients [SB11]. Since then, several other linearizations have been proposed for rational eigenproblems (see, for instance, [Güt+14]). It is therefore alluring to locally approximate $F(z) \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$ in Ω with a rational function $R(z)$ and then use the eigenpairs of $R(z)$ as the eigenpairs of $F(z)$ [Güt+14; Hoc17; EG19]. We can say that these methods are composed by two independent steps: the rational approximation of $F(z)$ and the solution of the eigenproblem related to $R(z) \approx F(z)$. Chapter 4 is mostly dedicated to the former part.

There we propose algorithms that upon successful execution always return rational approximations $R(z)$ such that their eigenvalues in Ω correspond to the eigenvalues of $F(z)$ in Ω.

Algorithms based on contour integrals

An elegant way to solve $G(\lambda)v = 0$ for $G(z) \in \mathcal{H}(\Omega_0, \mathbb{C}^{n\times n})$ is based on Keldysh's theorem, which was proved for polynomials in [Kel71] and in [GS71] for holomorphic functions. It states that for any function $G(z) \in \mathcal{H}(\Omega_0, \mathbb{C}^{n\times n})$, we can write

$$G(z)^{-1} = V(zI - J)^{-1}W^* + R(z),$$

for $z \in \Omega$, where $R(z) \in \mathcal{H}(\Omega, \mathbb{C}^{n\times n})$, V and W are right and left generalised eigenvector matrices, and J is a *Jordan matrix*, i.e., a block-diagonal matrix of Jordan blocks, whose eigenvalues are the eigenvalues of $G(z)$ in Ω. Together with Theorem 1.1 it follows that

$$\int_{\partial\Omega} g(z)G(z)^{-1}\,dz = Vg(J)W^*, \tag{1.5}$$

for any function $g(z) \in \mathcal{H}(\Omega, \mathbb{C})$. Several algorithms were hence developed from the previous result. Here we recall the Sakurai-Sugiura (SS) and its modifications [Asa+09; Asa+10; Che+17], Beyn's algorithm [Bey12; BEG20], and FEAST [Pol09; GMP18], among the most famous ones. The main advantage the authors usually underline is the incredible versatility they allow. First, they only require a black box form of $G(z)$, since nowhere in the algorithms an explicit knowledge of the matrix coefficients is needed. Further, they can both be used to retrieve all (or many) eigenvalues of medium-sized eigenproblems, or only few of them if $G(z)$ is large and sparse (and in that case $G(z)^{-1}$ is projected on a small subspace). In addition, they guarantee the discovery of all the eigenvalues in the target region Ω, as opposed to the Newton method and to the linearization ones, where we are computing the eigenpairs of an approximation. Finally, they are highly parallelisable: in fact, we can divide Ω in smaller subregions $\Omega_1, \dots, \Omega_s$ and each of the subproblems will be independent;

furthermore, when (1.5) is approximated with a quadrature rule, each linear system is independent from each other and thus it can be solved by a different processor.

We will deeply analyse this class of algorithms in Chapters 2 and 3, therefore we direct the reader to the specific introductions for further details.

1.3 APPLICATIONS

We already hinted that nonlinear eigenvalue problems lie at the heart of many technologies of our world. In this section we explore some examples from optics, electronics, and mechanics. All of them are included in the "Nonlinear Eigenvalue Problems" (NLEVP) library since version 4.0 [Bet+11; HNT19].

1.3.1 The buckling of a plane frame

When a solid structure, such as a column, a plane frame or a steel beam, is subjected to a heavy load, it may suddenly deform and change its shape. In structural engineering, this phenomenon is called *buckling*. Euler was among the first studying this subject. He proved that a slender column of length L compressed by a longitudinal force P maintains its shape until the load reaches

$$P_n = \frac{n^2 \pi E I}{L^2}, \qquad n = 1, 2, \ldots, \tag{1.6}$$

where E is elasticity coefficient of the column material and I is the moment of inertia. Nowadays, P_n is known as the n-th mode of Euler's critical load.

If a finite elements formulation is used, then the buckling problem can be stated as a generalised eigenvalue problem (see, for instance, [Ant11; BW73]).

Wittrick and Williams avoided this pathway and proposed an algorithm to find the eigenvalues and eigenvectors of a NEP which correspond to the critical load and the displacement components of the buckled structure [WW71; WW73]. In Figure 1.1

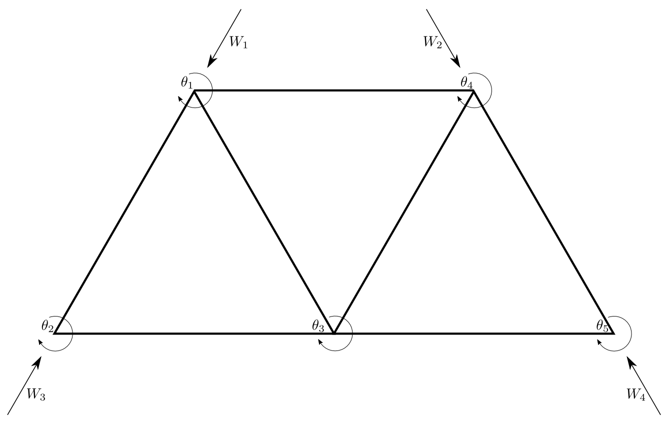


Figure 1.1: The scheme of the buckling problem in rigid-jointed triangulated shaped plane frame.

we show a rigid-jointed triangulated plane frame subject to symmetrical loads $W_i = \lambda P_1$ for $i = 1, \ldots, 4$, similar to the ones in [WW73, Section V]. We assume that the beams are inextensible, hence there are no translational displacements. We denote with θ_i, for $i = 1, \ldots, 5$ the rotations of the joints during the buckling: due to the symmetries in the problem, the buckling must be either symmetric (hence $\theta_3 = 0$) or antisymmetric. In the latter case, we can identify θ_4 with θ_1, and θ_5 with θ_2. Thanks to physical considerations the nonlinear eigenvalue problem becomes

$$F(\lambda) = \begin{bmatrix} f(\lambda)+10 & f(\lambda)g(\lambda) & 2 \\ f(\lambda)g(\lambda) & f(\lambda)+4 & 2 \\ 2 & 2 & 8 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = 0,$$

where $f(\lambda)$, $g(\lambda)$ are the stability functions defined by the relationships

$$f(\lambda)(1+g(\lambda)) = \frac{2\lambda^2 \sin\lambda}{\sin\lambda - \lambda\cos\lambda}, \qquad f(\lambda)(1-g(\lambda)) = 2\lambda\cot\lambda$$

which are sometimes denoted by $s(\lambda)$ and $c(\lambda)$ respectively [LC56]. The NLEVP library contains a simplified version of this problem, called `buckling_plate`.

1.3.2 Time-delay systems

Several physical systems need to be modelled through Delay-Differential Equations (DDE) of the form

$$\dot{x}(t) = A_0 x(t) + \sum_{k=1}^{s} A_k x(t - \tau_k). \tag{1.7}$$

For instance, every time we input a command in an electronic device, the electric signal takes some time to reach its destination; during a traffic jam, drivers do not immediately accelerate or brake when the car in front starts again or stops.

Substituting the sample solution $x(t) = \mathrm{e}^{\lambda t} v$ in (1.7) yields the nonlinear eigenvalue problem

$$(\lambda I - A_0 - \sum_{k=1}^{m} A_k \mathrm{e}^{-\lambda \tau_k}) v = 0. \tag{1.8}$$

Retrieving the eigenvalues of (1.8) gives us important information on the stability of the underlying physical system. See, for example, [Jar12; JM10] or [MN07] and the references therein. An interesting example is given by a semiconductor laser subject to a delayed phase-conjugate feedback caused by a reflective mirror, as schematised in Figure 1.2 [GW06, Section 5]. The equations of the model are

$$\begin{cases} \dfrac{1}{\tau}\dfrac{\mathrm{d}E}{\mathrm{d}t} = \dfrac{1}{2}\left[-\mathrm{i}G_N(N(t) - N_{\mathrm{sol}}) + \left(G(t) - \tau_p^{-1}\right)\right] E(t) + kE^*(t-1), \\ \dfrac{1}{2}\dfrac{\mathrm{d}N}{\mathrm{d}t} = \dfrac{I}{q} - \dfrac{N(t)}{\tau_e} - G(t)|E(t)|^2, \end{cases}$$

where $E(t)$ is the complex electric field, $N(t)$ represents the population inversion of the laser, and $G(t)$ is the nonlinear gain of the electric field, while the other parameters are physical constants. It is assumed that the feedback is quite weak, hence only a single delay term is considered. A linearization of $G(t)$ then leads to a 3-by-3 problem of the form of (1.8) with $m = 1$. The name of this problem in the NLEVP library is `laser`.

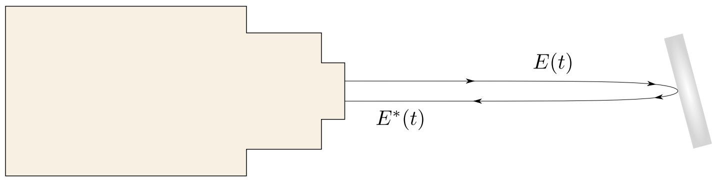


Figure 1.2: The phase-conjugate of the laser wave is reflected back by the mirror as a feedback.

1.3.3 Design of optical fibres

Optical fibres are omnipresent in current technologies and allow data transfer at much higher bandwidth than electrical cables. They are composed by a glass core, which act as a waveguide for the light, surrounded by a transparent material, the cladding, with a lower index of refraction, as depicted in Figure 1.3.

Huang, Bai, and Su considered an ideal cylindrical optical fibre, where the cladding radius R is much larger than the core radius R_c [HBS10]. In the cylindrical coordinate system (r, θ, z), Maxwell's equation for the guided wave function $f(r)$ becomes

$$\left[\frac{1}{r}\frac{\mathrm{d}}{\mathrm{d}r}\left(r\frac{\mathrm{d}}{\mathrm{d}r}\right) - \frac{m^2}{r^2} + (k^2(r) - k_{\mathrm{cl}}^2)\right] f = (\beta^2 - k_{\mathrm{cl}}^2) f, \tag{1.9}$$

where m is a positive integer, β is an unknown propagation constant, $k_{\mathrm{cl}} = 2\pi\eta_{\mathrm{cl}}/l$ is the wave number in the cladding, l is the light vacuum wavelength, and η_{cl} is the cladding refractive index; in addition, $k(r) = 2\pi\eta(r,l)/l$ is the wave number in the glass core, with $\eta(r,l)$ being the core refractive index.

Under the model assumptions, the guided wave function has the form

$$f(r) = aK_m(\mu r),$$

where a is an unknown constant, $\mu = \sqrt{\beta^2 - k_{\mathrm{cl}}^2}$, and K_m is the m-th order modified Bessel function of the second kind (see, for example, [AS64, Chapter 9]).

The boundary conditions

$$f(0) = 0, \qquad f(R) = \mu\frac{K_m'(\mu R)}{K_m(\mu R)}$$

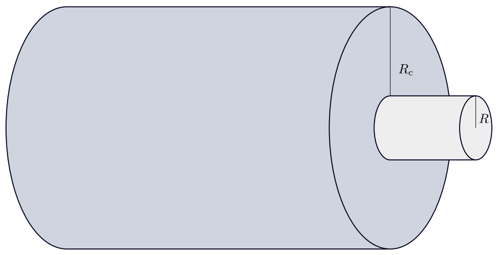


Figure 1.3: The core (light grey) and the cladding (dark grey) of an optical fiber.

along with a finite element method of $n+2$ equispaced points allow to reinstate (1.9) as the nonlinear eigenvalue problem

$$[A - \lambda I + s(\lambda) e_n e_n^T]v = 0,$$

where A is a symmetric tridiagonal matrix and

$$s(z) = \left(1 + \frac{1}{2n}\right)\sqrt{z}\frac{K_m'(n\sqrt{z})}{K_m(n\sqrt{z})}.$$

Finding the smallest positive eigenvalue λ is then equivalent to retrieving the constant β. This is the first step in studying the chromatic dispersion, which severely limits the transmission distances and causes signal distortion [Kar00]. The name of this problem in the NLEVP library is `fiber`.

1.3.4 Canyon particle

Solving the electronic transport in semiconductive material requires the study of the Poisson, Schrödinger, and transport equations. However, if the model assumes that the collisions of the electrons are negligible, then solving the single-particle Schrödinger equation with transmitting boundary conditions determines the current of the system [LK90; Van+14]. In the case of a system with contacts, the Schrödinger equation can be cast as the nonlinear eigenvalue problem

$$[-\nabla^2 + U(x)]\psi(x) = \lambda\psi(x). \tag{1.10}$$

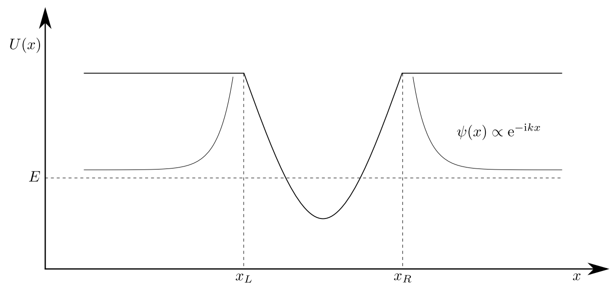


Figure 1.4: One dimensional scheme with contacts starting at $x = x_L$ and $x = x_R$.

$U(r)$ is the potential energy, while the eigenvalues λ are the bound states, and the eigenvectors the wave functions. In Figure 1.4 we schematise the one dimensional problem. A finite element discretization of (1.10) for the two-dimensional case yields

$$\left(H - \lambda I + \sum_{j=1}^{n_z} e^{i\sqrt{m(\lambda - \alpha_j)}} L_k U_k^* \right) v = 0,$$

where $H \in \mathbb{R}^{n \times n}$ is a symmetric matrix, m is the mass of the particle, n_z is the number of mesh points, α_j are physical parameters, and L_k, U_k are low-rank matrices. A detailed description of this problem can be found in [Van+14; Güt+14], while the NLEVP library has a practical implementation named `canyon_particle`.

1.4 STRUCTURE OF THE THESIS AND CONTRIBUTIONS

The thesis is structured as follows. Chapter 1 was dedicated to the introduction of the work and to a review of the prerequisites needed to understand the thesis.

Chapters 2, 3, and 4 mirror Section 1.2.2. In Chapter 2 we generalise the theory of contour integral algorithms from holomorphic to meromorphic functions and we explain how the presence of the poles influences the convergence of the methods. Our attention is mostly on small- to medium-sized eigenvalue problems, hence particular focus is given to the case where the number of eigenvalues in Ω exceeds n.

Chapter 3 is dedicated to more practical considerations concerning contour integrals and to our implementation of a contour eigensolver. First, we show the improvements and the new features between the NLEVP 3.0 and its current version NLEVP 4.1. This library has been a fundamental tool to understand nonlinear eigenproblems and we are sure the same will be true for other researchers. In addition, contour solvers require the choice of multiple parameters (for example, the number of quadrature points), which are often left to the final user. There we propose multiple ways to implement an algorithm which automatically chooses these parameters, hence allowing the user to only provide the region Ω and function $F(z)$ in black-box form. Finally, we show that combining the Newton method or a similar refinement with the core of the contour solver creates a very robust algorithm.

Chapter 4 is dedicated to eigensolvers based on linearisations. More precisely, we focus primarily on the first step of these algorithms, i.e., the approximation of $F(z)$ with a rational function $R(z)$. First, we show how the backward error of the eigenpairs of $F(z)$ is related to the backward error of the eigenpairs of $R(z)$, provided that $R(z)$ is a "good approximation" to $F(z)$. This theoretical result allows us to propose two robust algorithms to return $R(z)$, one for $F(z)$ given in split form, another for $F(z)$ in black-box form.

At the end of the chapter we test our algorithms with the state of the art ones on the NLEVP collection and we show their approximations are better or at least as good as the previous ones. Finally, we also compare them with the contour algorithm proposed in Chapter 3.

Finally, Chapter 5 stands apart from the preceding ones. There we explore tropical linear algebra, a relatively new branch of mathematics. Recent results by Sharify et al. [Sha11] found a quantitative relationship between the roots of a scalar polynomial and the so-called tropical roots of it. These results were later generalised to matrix polynomials [NST15], and hence tropical roots where used as an initial approximation for their eigenvalues or as a way to scale the polynomial such as other algorithms, like POLYEIG, could perform better [TV21; VT18]. We expand the theory from polynomials to Laurent series and we show that similar results hold true in these general settings as well.

2 CONTOUR INTEGRAL METHODS FOR MEROMORPHIC EIGENVALUE PROBLEMS

2.1 INTRODUCTION

In this chapter we analyse in greater depth contour integral solvers (or contour solvers, for brevity) for nonlinear eigenvalue problems on a target region Ω. Since the beginning of the XXI century, researchers in numerical linear algebra showed great interest in this class of algorithms.

In 2003 Sugiura and Sakurai were the first to develop a contour solver for the linear eigenvalue problem [SS03]. In 2009 Polizzi implemented the FEAST algorithm for the symmetric eigenvalue problem [Pol09]. It is now at its fourth release and since 2018 is able to tackle nonlinear problems [GMP18]. Still in 2009, Asakura, Sakurai et al. updated the Sugiura–Sakurai (SS) [Asa+09] method to solve nonlinear problems. They named it block-SS and it is based on the Smith decomposition. In 2012 Beyn introduced his algorithm, which is based on Keldysh's theorem and considered also the case where the number of eigenvalues in Ω is larger than n [Bey12]. In 2013 Yokota and Sakurai proposed the SS-RR (Rayleigh–Ritz) for large scale systems, whose core idea is projecting the problem on a smaller subspace and then use one of the other nonlinear solvers for it [YS13]. In a recent ArXiv preprint, Krenner and Polizzi introduced a similar idea, where they iteratively use FEAST on large problems to project them on smaller equivalent ones, which are then solved by Beyn's algorithm [BP20]. In the meantime, Huang, Su, et al. created the Recursive Integral Algorithm (RIM) for the generalised eigenvalue problem [Hua+16]. As opposed to the other algorithms mentioned above, RIM uses Cauchy's Integral Theorem 1.1 and its core is very easy to implement. It was mainly aimed at linear eigenvalue problems, but it can be applied to the holomorphic case as well.

All the previous research concerning contour integrals required the function $G(z)$ to be holomorphic. The main goal of this chapter is developing a solid theoretical background of contour solvers for meromorphic eigenvalue problems (MEP). There are two main reasons which serve as a backstory of this work. The first was a comment by Beyn in the survey paper by Güttel and Tisseur, where he stated that this kind of algorithms should work for this larger class of problems [GT17, Section 5.5]. Preliminary numerical experiments proved this to be right, but it was clear that there was something deeper behind it. The second, probably a bit more naive, is the fact that rational eigenvalue problems (REP) are not included in the class of holomorphic eigenvalue problems (HEP). In Section 1.2 we saw the hierarchy of these classes. In the first three cases, they were "exact" generalisations: a linear eigenvalue problem is also polynomial, which in turn is rational; however, a rational problem is not generally holomorphic, because it may have poles in Ω. Hence, developing the theory for the MEP will address this asymmetry.

The chapter is structured as follows. In Section 2.2 we recall the basic theory needed to state Keldysh's theorem and we introduce new definitions that will allow to study the meromorphic eigenvalue problem through contour integrals. The main one is the *holomorphization* of a meromorphic function $F(z) \in \mathcal{M}(\Omega, \mathbb{C}^{n\times n})$. This purely theoretical tool allows us to understand how the poles of $F(z)$ influence contour solver algorithms. Furthermore, we propose a way to compute the backward error of eigenpairs for functions given in black-box form. In Section 2.3 we give a brief review of the Recursive Integral Method in the holomorphic case and we calculate its probabilistic computational cost, which was missing in the current literature. Section 2.4 is dedicated to Beyn's algorithm: first, we review its original description, then we explain its Loewner interpretation, and finally we expand it to meromorphic functions, both for simple eigenvalues and in the general case. There is no real conclusion section, because in Chapter 3 we continue developing these themes under an algorithmic point of view.

2.2 THEORETICAL BACKGROUND

In Section 1.2.1 we hinted how a nonlinear problem may have generalised eigenvectors which are neither linearly independent from the corresponding eigenvector nor from eigenvectors of other eigenvalues. Even though in many real-life applications it is assumed that all the eigenvalues are either simple or semisimple, hence removing the burden of generalised eigenvectors, the analysis under these settings is necessary to understand how contour integrals can be effectively used for meromorphic eigenvalue problems.

In the undergrad course of linear algebra we have learned that for a given eigenvalue λ of a matrix $A \in \mathbb{C}^{n\times n}$, the sequence of vectors $v_0, \dots, v_{m-1}$ is called a *Jordan chain* if

$$(A-\lambda I)v_0 = 0, \quad (A-\lambda I)v_1 = v_0, \quad \dots \quad (A-\lambda I)v_{m-1} = v_{m-2}, \tag{2.1}$$

where v_0 is, in fact, an eigenvector, while v_j are the generalised eigenvectors of the Jordan chain associated to v_0. In 1968 Trofimov introduced the concept of *root functions* $v(z) = \sum_{j=0}^{m-1}(z-\lambda)^j v_j$ and pointed out that the equalities of (2.1) are equivalent to λ being a root of multiplicity at least m of $(A - zI)v(z)$ [Tro68]. This idea can be simply generalised to the nonlinear case:

Definition 2.1 ([GT17, Definition 2.3])**.** Consider $G(z) \in \mathcal{H}(\Omega_0, \mathbb{C}^{n\times n})$ and let $\lambda \in \Omega$ being an eigenvalue of $G(z)$.

1. A function $v \in \mathcal{H}(\Omega, \mathbb{C}^n)$ is said to be a *root function for* $G(z)$ *at* λ if $v(\lambda) \neq 0$ and $G(\lambda)v(\lambda) = 0$. The multiplicity of the root λ of $G(z)v(z)$ is denoted by $s(v)$.

2. A tuple $(v_0, \dots, v_{m-1})$ is called a *Jordan chain for* $G(z)$ *at* λ if

$$v(z) = \sum_{k=0}^{m-1}(z-\lambda)^k v_k$$

is a root function for $G(z)$ at λ and $s(v) \geqslant m$.

3. For any eigenvector v_0 of λ, the *rank of* v_0 is defined as

$$r(v_0) = \max\{s(v) : v \text{ is a root function for } G(z) \text{ at } \lambda \text{ with } v(\lambda) = v_0\}.$$

4. A system of vectors in $\mathbb{C}^n$

$$V_\lambda := (v_k^j : 0 \leqslant k \leqslant m_j - 1, 1 \leqslant j \leqslant d)$$

is called a *complete system of Jordan chains* if the following conditions are met:

   a) $d = \dim(\text{null}(G(\lambda))$ is the geometric multiplicity of λ and $\{v_0^1, v_0^2, \dots, v_0^d\}$ is a basis of $\text{null}(G(\lambda))$.

   b) For $1 \leqslant j \leqslant d$, $(v_0^j, v_1^j, \dots, v_{m_j-1}^j)$ is a Jordan chain for $G(z)$ at λ.

   c) For $1 \leqslant j \leqslant d$, $m_j = \max\{r(v_0) : v_0 \in \text{null}(G(\lambda)) \setminus \text{span}\{v_0^\nu : 1 \leqslant \nu < j\}\}$.

The previous definition has got a lot to unpack. First, one can show that a complete system of Jordan chains always exists and that $\sum_{j=1}^d m_j$ is equal to the algebraic multiplicity of λ [MM03, Propositions 1.6.4, 1.8.4]. In addition, we call m_1 the *index* of λ, while all the numbers $m_1 \geqslant m_2 \geqslant \cdots \geqslant m_d$ are the *partial multiplicities* of λ. It follows that an eigenvalue is semisimple if $m_1 = \cdots = m_d = 1$, while it is simple if $d = 1$, and thus V_λ will be just the tuple of the eigenvector(s) of λ.

Definition 2.1 covers the case of the right generalised eigenvectors of $G(z)$. Clearly, the left complete system of Jordan chains W_λ is defined in the same way for $G^*(z)$. Furthermore, if we set V_λ, then there exists a unique canonical W_λ which satisfies the following conditions.

Theorem 2.1 ([MM03, Theorem 1.6.5]). *Let $\lambda \in \Omega$ be an eigenvalue of $G(z) \in \mathcal{H}(\Omega_0, \mathbb{C}^{n\times n})$ and V_λ be*

$$V_\lambda = (v_k^j : 0 \leqslant k \leqslant m_j - 1, 1 \leqslant j \leqslant d)$$

a complete system of Jordan chains as defined in 2.1. Then there exists a unique system of Jordan chains W_λ for $G^*(z)$

$$W_\lambda = (w_k^j : 0 \leqslant k \leqslant m_j - 1, 1 \leqslant j \leqslant d)$$

such that each eigenvector w_0^j has rank $r(w_0^j) = m_j$ and satisfies

$$\Psi_\lambda(G,k,i,j) = \sum_{\alpha=0}^{k} \sum_{\beta=1}^{m_i} w_{k-\alpha}^{j*} \frac{G^{(\alpha+\beta)}(\lambda)}{(\alpha+\beta)!} v_{m_i-\beta}^{i} = \delta_{ij}\delta_{0k}, \tag{2.2}$$

for $0 \leqslant k \leqslant m_j - 1$, and $1 \leqslant i, j \leqslant d$, where δ_{ij} is the Kronecker delta.

The normalisation conditions $\Psi_\lambda(G,k,i,j)$ in Theorem 2.1 are very complicated in the general case, however they become much more reasonable when the eigenvalue λ is semisimple, i.e., when all the partial multiplicities m_i are equal to 1. If that is the case, then the following corollary holds.

Corollary 2.2. *Let $\lambda \in \Omega$ be a semisimple eigenvalue of $G \in \mathcal{H}(\Omega_0, \mathbb{C}^{n\times n})$, and let $\{v_1, \dots, v_d\}$ be a basis of* $\mathrm{null}(G(\lambda))$. *Then there exists a unique basis $\{w_1, \dots, w_d\}$ of* $\mathrm{null}(G^*(\lambda))$ *such that*

$$w_i^* G'(\lambda) v_j = \delta_{ij},$$

where δ_{ij} is the Kronecker delta.

Armed with these results, we can now state Keldysh's decomposition, the fundamental theorem for contour integral algorithms.

2.2.1 Keldysh decomposition

A very useful factorization that introduces Keldysh's theorem is the Smith form. It was first developed by Smith in 1861 to solve linear systems of Diophantine equations [SS61], while Frobenius later extended it to matrix polynomials [Fro79]. We here report the general form found in [KM99] and [GT17], which uses the notion of *unimodular matrix-valued functions*: we say $P(z) \in \mathcal{H}(\Omega, \mathbb{C}^{n\times n})$ is unimodular if $\det P(z) \in \mathbb{C}$ is a nonzero constant or, equivalently, if $P^{-1}(z) \in \mathcal{H}(\Omega, \mathbb{C}^{n\times n})$.

Theorem 2.3 (Smith form). *Consider $G(z) \in \mathcal{H}(\Omega_0, \mathbb{C}^{n\times n})$ and let $\lambda_1, \dots, \lambda_s$ be its distinct eigenvalues in Ω with partial multiplicities $m_{i,1} \geqslant m_{i,2} \geqslant \cdots \geqslant m_{i,d_i}$. Then there exist two unimodular matrix-valued functions $P(z), Q(z) \in \mathcal{H}(\Omega, \mathbb{C}^{n\times n})$ such that*

$$P(z)G(z)Q(z) = D(z),$$

where $D(z) = \operatorname{diag}(\delta_1(z), \dots, \delta_n(z))$ is a diagonal matrix with entries

$$\delta_j(z) = h_j(z)\prod_{i=1}^{s}(z-\lambda_i)^{m_{i,j}}, \qquad j = 1, \dots, n,$$

where each $h_j \in \mathcal{H}(\Omega, \mathbb{C})$ does not have zeros in Ω and $m_{i,j} = 0$ when $j > d_i$.

Now, denote by $p_j(z)$ and $q_j(z)$ the j-th column of the matrices $P(z)$, $Q(z)$ respectively. Then we can write

$$G(z)^{-1} = \sum_{j=1}^{n} \delta_j(z)^{-1} q_j(z) p_j(z)^*,$$

and if we expand every single term of this sum as a Laurent series in some neighbourhood $\mathcal{U}$ of λ_1, we have

$$G(z)^{-1} = \sum_{j=1}^{d_1}\sum_{k=1}^{m_{1,j}} S_{1,j,k}(z-\lambda_1)^{-k} + R_1(z), \qquad z \in \mathcal{U}\backslash\{\lambda_1\}, \tag{2.3}$$

where $R_1(z) \in \mathcal{H}(\mathcal{U})$, and $S_{1,j,k} \in \mathbb{C}^{n\times n}$. If we expand $R_1(z)$ recursively around λ_i for $i = 2, \dots, s$, identity (2.3) becomes

$$G(z)^{-1} = \sum_{i=1}^{s}\sum_{j=1}^{d_i}\sum_{k=1}^{m_{i,j}} S_{i,j,k}(z-\lambda_i)^{-k} + R(z), \qquad z \in \mathcal{U}\backslash\{\lambda_1\}, \tag{2.4}$$

with $R(z)$ being holomorphic on the entire set Ω.

The strength of Keldysh's theorem lies in the characterisation of the matrices $S_{i,j,k}$ in terms of generalised eigenvectors[GS71; GT17]. Let us denote the complete right and left Jordan chains for an eigenvalue λ_i

$$(v_k^{ij} : 0 \leqslant k \leqslant m_{i,j} - 1, 1 \leqslant j \leqslant d_i), \qquad (w_k^{ij} : 0 \leqslant k \leqslant m_{i,j} - 1, 1 \leqslant j \leqslant d_i)$$

and let V_{ij}, W_{ij} be the matrices

$$V_{ij} = \begin{bmatrix} v_0^{ij} & v_1^{ij} & \dots & v_{m_{i,j}-1}^{ij} \end{bmatrix}, \qquad W_{ij} = \begin{bmatrix} w_{m_{i,j}-1}^{ij} & w_{m_{i,j}-2}^{ij} & \dots & w_0^{ij} \end{bmatrix}. \tag{2.5}$$

If we define the Jordan blocks

$$J_{ij} = \begin{bmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{bmatrix} \in \mathbb{C}^{m_{i,j} \times m_{i,j}} \tag{2.6}$$

then we can state Keldysh's theorem in the following form.

Theorem 2.4 (Keldysh). *Consider $G(z) \in \mathcal{H}(\Omega_0, \mathbb{C}^{n\times n})$ under the hypotheses of Theorem 2.3 and define $\overline{m} := \sum_{i=1}^{s} \sum_{j=1}^{d_i} m_{i,j}$. Then there exist two $n \times \overline{m}$ matrices V, W, and a $\overline{m} \times \overline{m}$ Jordan matrix J such that*

$$G(z)^{-1} = V(zI - J)^{-1}W^* + R(z)$$

for a function $R(z) \in \mathcal{H}(\Omega, \mathbb{C}^{n\times n})$, where

$$\begin{aligned} J &= \operatorname{diag}(J_1, \cdots, J_s), & J_i &= \operatorname{diag}(J_{i1}, \cdots, J_{id_i}), \\ V &= [V_1, \cdots, V_s], & V_i &= [V_{i1}, \cdots, V_{id_i}], \\ W &= [W_1, \cdots, W_s], & W_i &= [W_{i1}, \cdots, W_{id_i}], \end{aligned}$$

and V_{ij}, W_{ij}, J_{ij} defined in (2.5–2.6).

Hypotheses 2.1. From now on we will constantly recall the spectral hypotheses of Theorems 2.3 and 2.4. We summarise them here in order to avoid unnecessary and distracting repetitions. If not specified otherwise, $G(z) \in \mathcal{H}(\Omega_0, \mathbb{C}^{n\times n})$ has $\lambda_1, \dots, \lambda_s$ distinct eigenvalues in Ω with partial multiplicities $m_{i,1} \geqslant m_{i,2} \geqslant \cdots \geqslant m_{i,d_i}$. We set $\overline{m} := \sum_{i=1}^{s} \sum_{j=1}^{d_i} m_{i,j}$ and we name the matrices $V, W \in \mathbb{C}^{n\times \overline{m}}$ the *eigenvector matrices* of $G(z)$ (in Ω) and V, W, J the *spectral matrices* of $G(z)$ (in Ω).

We can see Keldysh's theorem as a generalisation of the Jordan canonical form of a matrix A in the nonlinear case. Indeed, if $A = VJV^{-1}$ is the Jordan decomposition, then $G(z)^{-1} := (zI - A)^{-1} = V(zI - J)^{-1}V^{-1}$, where $R(z) \equiv 0$ and $W^* = V^{-1}$. In addition, if we have stricter hypotheses on the nature of the eigenvalues, then Theorem 2.4 becomes easier to state and understand. For instance, this is the version chosen by Van Barel and Kravanja in [VK16].

Corollary 2.5 (Keldysh's theorem for simple eigenvalues). *Consider $G(z) \in \mathcal{H}(\Omega_0, \mathbb{C}^{n\times n})$ and let $\lambda_1, \dots, \lambda_s$ be its distinct, simple eigenvalues in Ω. Then there exists a holomorphic function $R \in \mathcal{H}(\Omega, \mathbb{C}^{n\times n})$ such that*

$$G(z)^{-1} = \sum_{j=1}^{s} \frac{v_j w_j^*}{z - \lambda_j} + R(z),$$

where $v_1, \dots, v_s$ and $w_1, \dots, w_s$ are right and left eigenvectors satisfying $w_j^* G'(\lambda_j) v_j = 1$.

As anticipated, we want to apply contour integral algorithms to meromorphic functions. In order to do so, we have to slightly generalise the theorems and corollary described above.

First of all, let us fix some settings. Let $F \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$ denote a regular meromorphic matrix-valued function with $\lambda_1, \dots, \lambda_s$ distinct eigenvalues in Ω and $\xi_1, \dots, \xi_r$ distinct poles in Ω. In addition, we assume that no eigenvalues and no poles lie on $\partial\Omega$. As written in Definition 1.5 of nonlinear eigenvalue problem, we do not consider the degenerate case where a pole coincides with an eigenvalue, such as for

$$F(z) = \begin{bmatrix} z^{-1} & (z-3)^{-1} \\ 0 & z(z-1) \end{bmatrix}, \tag{2.7}$$

where $z = 0$ is both an eigenvalue and a pole. Note that $\det F(z) = z - 1$, hence a pole of $F(z)$ does not need to be a pole of $\det F(z)$, as it happens for $z = 0$ or $z = 3$ in (2.7).

As anticipated in Section 1.1, it is always possible to find a polynomial $g(z) \in \mathbb{C}[z]$ such that $G(z) := g(z)F(z)$ is holomorphic (at least) in Ω. It is important to point out that the original poles of $F(z)$ may or may not be eigenvalues of $G(z)$. For example, consider

$$F_1(z) = \begin{bmatrix} z^{-2}(z-1) & 3 \\ 0 & z^{-2}(z-2) \end{bmatrix}, \qquad F_2(z) = \begin{bmatrix} z^{-2}(z-1) & (z-2)^{-3} \\ 0 & z^{-1} \end{bmatrix}. \tag{2.8}$$

Then we can define $g_1(z) = z^2$ and $g_2(z) = z^2(z-2)^3$, so that both $G_k(z) = g_k(z)F(z)$ for $k = 1, 2$ are holomorphic. However, while $G_1(z)$ has the same eigenvalues of $F_1(z)$, $\xi_1 = 0$, $\xi_2 = 2$ are eigenvalues of $G_2(z)$, but not eigenvalues of $F_2(z)$. It makes sense to introduce the following concept.

Definition 2.2 (Spurious eigenvalues). A pole ξ of $F(z)$ of multiplicity c is a *spurious eigenvalue* of $F(z)$ if it is an eigenvalue of $G(z) := (z-\xi)^c F(z)$. In addition, the algebraic and geometric multiplicities of the spurious eigenvalue ξ of F are defined as the algebraic and geometric multiplicities of ξ as eigenvalue of $G(z)$.

Remark 2.1. Definition 2.2 is inspired by the definition of eigenvalues at infinity for polynomial eigenvalue problems, where we say that $\lambda = \infty$ is an eigenvalue of a degree c matrix polynomial $P(z)$ if $\lambda = 0$ is an eigenvalue of the reversal $z^c P(z^{-1})$.

Unfortunately, we cannot state many properties about spurious eigenvalues. First, as witnessed in (2.8), the original poles may or may not become spurious eigenvalues. Furthermore, there is no clear relationship between their multiplicity as poles of $F(z)$ and their algebraic or geometric multiplicity as spurious eigenvalues. For example, consider $\xi = 0$ and

$$F_1(z) = \begin{bmatrix} z^{-2}(z-1) & 0 & z^{-2} \\ 0 & z^{-2}(z-2) & 0 \\ 0 & 0 & z^{-2}(z-3) \end{bmatrix}, \qquad F_2(z) = \begin{bmatrix} z-1 & 0 & z^{-1} \\ 0 & z-2 & 0 \\ 0 & 0 & z-3 \end{bmatrix}.$$

Then ξ is a double pole for $F_1(z)$, but is not a spurious eigenvalue for $G_1(z) = z^2 F_1(z)$. On the other hand, ξ is a single pole for $F_2(z)$, but it is a spurious eigenvalue of $G_2(z) = z^2 F_2(z)$ with algebraic multiplicity equal to 3 and geometric multiplicity equal to 2, hence $G_2(z)$ will have a 2×2 Jordan block with eigenvalue 0.

It is not too difficult to see that we can retrieve the spectral properties of the spurious eigenvalues of $F(z)$ by considering an auxiliary function $G(z)$, which is holomorphic in Ω, as illustrated in the following lemma.

Lemma 2.6 (Holomorphization of a meromorphic function). *Let $F(z) \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$ be a regular matrix-valued function with $\lambda_1, \dots, \lambda_s$ distinct eigenvalues in Ω and $\xi_1, \dots, \xi_r$ distinct poles in Ω. Let $e_i \in \mathbb{R}^n$ be the ith column of the identity basis and write*

$$F(z) = \sum_{i,j=1}^{n} f_{ij}(z) e_i e_j^T. \tag{2.9}$$

In addition, define c_k as

$$c_k := \max_{1 \leqslant i,j \leqslant n} \{\textit{multiplicity of the pole } \xi_k \textit{ in } f_{ij}(z)\}, \qquad k = 1, \dots, r.$$

Then there exists a unique monic polynomial $g(z) = \prod_{k=1}^{r} (z - \xi_k)^{c_k}$ with $c_k > 0$ such that $G(z) := g(z)F(z) \in \mathcal{H}(\Omega, \mathbb{C}^{n\times n})$, and we say $G(z)$ is the holomorphization *of $F(z)$. Furthermore, the eigenvalues of $G(z)$ are all and only the eigenvalues and spurious eigenvalues of $F(z)$ and they have the same algebraic and geometric multiplicities.*

Proof. First, we need to prove that $G(z)$ is holomorphic. In order to do so, note that (2.9) is unique, because it is simply the entry-wise form of $F(z)$. In addition, if ξ is a pole of $f_{ij}(z)$, then it is a pole of $F(z)$, because the matrices $e_i e_j^T$ are linearly independent. This implies that c_k is the multiplicity of ξ as a pole of $F(z)$ and that $G(z)$ is holomorphic. Finally, we need to show that the eigenvalues of $G(z)$ have the same spectral properties as the eigenvalues and spurious eigenvalues of $F(z)$. First, set $G_0(z) = F(z)$ and $G_k(z) = (z - \xi_k)^{c_k} G_{k-1}(z)$ for $k = 1, \dots, r$. It is not difficult to see that $G_k(z)$ has the same eigenvalues of $G_{k-1}(z)$ with the same properties and will have at most ξ_k as a new eigenvalue if, by definition, ξ_k itself is a spurious eigenvalue of $G_{k-1}(z)$, since $\xi_k \notin \Lambda(G_{k-1})$. The result then simply follows from $G(z) = G_r(z)$. □

Note that $G(z)$ is not generally holomorphic on all Ω_0. An easy example of why this cannot be true is given by $F(z) = \sin\left(z^{-1}\right)^{-1}$, with $\Omega_0 = \mathcal{D}(1,1)$ and $\Omega = \mathcal{D}(1,0.9)$. The poles of $F(z)$ are $k^{-1}\pi^{-1}$ for $k \geqslant 1$ and they accumulate in zero. Hence, we can find a polynomial $g(z)$ such that $g(z)F(z)$ is holomorphic in Ω, but not in Ω_0. We can now state Keldysh's theorem for meromorphic functions in a precise way.

Theorem 2.7 (Keldysh for meromorphic functions). *Consider $F(z) \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$ and let $G(z) := g(z)F(z) \in \mathcal{H}(\Omega, \mathbb{C}^{n\times n})$ be the holomorphization of $F(z)$, with $g(z) \in \mathbb{C}[z]$ defined in Lemma 2.6. Let $m_{\xi_i,1} \geqslant m_{\xi_i,2} \geqslant \cdots \geqslant m_{\xi_i,d_{\xi_i}}$ be the partial multiplicities of ξ_i, with $d_{\xi_i} = 0$ if ξ_i is not a spurious eigenvalue and define*

$$\overline{m}_\lambda = \sum_{i=1}^{s}\sum_{j=1}^{d_i} m_{i,j}, \quad \overline{m}_\xi = \sum_{i=1}^{r}\sum_{j=1}^{d_{\xi_i}} m_{\xi_i,j}.$$

Then there exist two pairs of matrices, $\check{V}, \widehat{W} \in \mathbb{C}^{n\times\overline{m}_\lambda}$, $\widetilde{V}, \widetilde{W} \in \mathbb{C}^{n\times\overline{m}_\xi}$, and two matrices $\widehat{J} \in \mathbb{C}^{\overline{m}_\lambda\times\overline{m}_\lambda}$, $\widetilde{J} \in \mathbb{C}^{\overline{m}_\xi\times\overline{m}_\xi}$ such that

$$F(z)^{-1} = g(z)(V(zI - J)^{-1}W^* + R(z)),$$

with $R(z) \in \mathcal{H}(\Omega, \mathbb{C}^{n\times n})$, where

$$V = \begin{bmatrix} \check{V} & \widetilde{V} \end{bmatrix}, \qquad J = \begin{bmatrix} \widehat{J} & \\ & \widetilde{J} \end{bmatrix}, \qquad W = \begin{bmatrix} \widehat{W} & \widetilde{W} \end{bmatrix},$$

$$\check{V} = \begin{bmatrix} \check{V}_1 & \cdots & \check{V}_s \end{bmatrix}, \quad \widehat{J} = \begin{bmatrix} \widehat{J}_1 & & \\ & \ddots & \\ & & \widehat{J}_s \end{bmatrix}, \quad \widehat{W} = \begin{bmatrix} \widehat{W}_1 & \cdots & \widehat{W}_s \end{bmatrix},$$

$$\check{V}_i = \begin{bmatrix} \check{V}_{i1} & \cdots & \check{V}_{id_i} \end{bmatrix}, \quad \widehat{J}_i = \begin{bmatrix} \widehat{J}_{i1} & & \\ & \ddots & \\ & & \widehat{J}_{id_i} \end{bmatrix}, \quad \widehat{W}_i = \begin{bmatrix} \widehat{W}_{i1} & \cdots & \widehat{W}_{id_i} \end{bmatrix},$$

and $\check{V}_{ij}$, $\widehat{W}_{ij}$, $\widehat{J}_{ij}$ are defined in (2.5–2.6), with the generalised eigenvectors satisfying the normalisation conditions $\Psi_\lambda(G,\cdot,\cdot,\cdot)$ (2.2). The matrices $\widetilde{V}$, $\widetilde{W}$ and $\widetilde{J}$ are partitioned accordingly.

Proof. Without loss of generality, we can assume that all the poles ξ_i are spurious eigenvalues, so that $d_{\xi_i} \geqslant 1$. If that were not the case, we could just consider a subset of the poles. Keldysh's theorem 2.4 applied to $G(z)$ yields

$$F(z)^{-1} = g(z)V(zI - J)^{-1}W^* + g(z)R(z), \tag{2.10}$$

where $R \in \mathcal{H}(\Omega)$. Up to a permutation, the matrices J, V, and W have the block structure

$$\begin{aligned} J &= \operatorname{diag}(\widehat{J}_1, \dots, \widehat{J}_s, \widetilde{J}_1, \dots, \widetilde{J}_r), \\ V &= \begin{bmatrix} \check{V}_1 & \dots & \check{V}_s & \widetilde{V}_1 & \dots & \widetilde{V}_r \end{bmatrix}, \\ W &= \begin{bmatrix} \widehat{W}_1 & \dots & \widehat{W}_s & \widetilde{W}_1 & \dots & \widetilde{W}_r \end{bmatrix}, \end{aligned}$$

where $\check{V}_k$, $\widehat{W}_k$, and $\widehat{J}_k$ ($\widetilde{V}_k$, $\widetilde{W}_k$, and $\widetilde{J}_k$, respectively) are defined in Theorem 2.4 for λ_k (ξ_k, respectively) and each complete system of Jordan chain satisfies the normalisation conditions $\Psi_{\lambda_k}(G, \cdot, \cdot, \cdot)$ (2.2). We can write

$$V = \begin{bmatrix} \check{V} & \widetilde{V} \end{bmatrix}, \qquad W = \begin{bmatrix} \widehat{W} & \widetilde{W} \end{bmatrix}, \qquad J = \begin{bmatrix} \widehat{J} & \\ & \widetilde{J} \end{bmatrix}, \tag{2.11}$$

where $\check{V}$, $\widehat{W}$ are $n \times \overline{m}_\lambda$ matrices, while $\widehat{J}$ is a $\overline{m}_\lambda \times \overline{m}_\lambda$ Jordan matrix. Therefore, substituting (2.10) in (2.11) yields the result. □

Corollary 2.8 (Keldysh for meromorphic functions with simple eigenvalues)**.** *We consider $F(z) \in \mathcal{M}(\Omega_0, \mathbb{C}^{n \times n})$ and let $\lambda_1, \dots, \lambda_s$ be its distinct, simple eigenvalues in Ω and $\xi_1 \dots, \xi_r$ be its distinct poles. Let $g(z)$, $G(z)$, $m_{\xi_i,j}$, $\widetilde{V}$, $\widetilde{J}$, $\widetilde{W}$, $R(z)$ be defined as in Theorem 2.7. Then*

$$F(z)^{-1} = \sum_{k=1}^{s} \frac{g(z)\check{v}_k\check{w}_k^*}{z - \lambda_k} + g(z)\widetilde{V}(zI - \widetilde{J})^{-1}\widetilde{W}^* + g(z)R(z)$$

where $\check{v}_1, \dots, \check{v}_s$ and $\check{w}_1, \dots, \check{w}_s$ be respectively the right and left eigenvectors such that $\check{w}_k^* G'(\lambda_k)\check{v}_k = 1$ for $j = 1, \dots, s$.

Table 2.1: Summary of the most important symbols used in this chapter.

Symbol	Explanation
$F(z)$	The meromorphic function in $\mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$
s, r	Number of distinct eigenvalues and distinct poles
λ_i	Eigenvalue of $F(z)$ in Ω
ξ_i	Pole of $F(z)$ in Ω
c_i	Pole multiplicity of ξ_i
$g(z)$, $G(z)$	$g(z) = \prod_{i=1}^{r}(z-\xi_i)^{c_i}$, $G(z) = g(z)F(z) \in \mathcal{H}(\Omega, \mathbb{C}^{n\times n})$
d_i	Geometric multiplicity of λ_i
d_{ξ_i}	Geometric multiplicity of ξ_i as spurious eigenvalue; $d_{\xi_i} = 0$ if ξ_i not a spurious eigenvalue
$m_{i,j}$	$m_{i,1} \geqslant \cdots \geqslant m_{i,d_i}$ partial multiplicities of λ_i
$m_{\xi_i,j}$	$m_{\xi_i,1} \geqslant \cdots \geqslant m_{\xi_i,d_{\xi_i}}$ partial multiplicities of ξ_i as spurious eigenvalue
$\overline{m}_\lambda$, $\overline{m}_\xi$	$\sum_{i=1}^{s}\sum_{j=1}^{d_i} m_{i,j}$, $\sum_{i=1}^{s}\sum_{j=1}^{d_{\xi_i}} m_{\xi_i,j}$
$\overline{m}$	$\overline{m}_\lambda + \overline{m}_\xi$
P	Probing matrix in contour integrals, $P \in \mathbb{C}^{n\times p}$
m	Number of moments in Hankel matrix $B_0^{[m]}$

We conclude this section with the standard notation and settings about meromorphic functions, both as a paragraph and as a table. We will constantly recall these hypotheses, mostly in Section 2.4.1.

Hypotheses 2.2. If not specified otherwise, $F(z) \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$ has $\lambda_1, \ldots, \lambda_s$ distinct eigenvalues and $\xi_1, \ldots, \xi_r$ distinct poles in Ω. We let $g(z) = \prod_{i=1}^{r}(z-\xi_i)^{c_i}$ and $G(z) := g(z)F(z) \in \mathcal{H}(\Omega, \mathbb{C}^{n\times n})$ be the polynomial and holomorphization defined in Lemma 2.6. Each pole ξ_i has a pole multiplicity equal to c_i and we assume that all the poles are spurious eigenvalues. We denote with $m_{i,1} \geqslant m_{i,2} \geqslant \cdots \geqslant m_{i,d_i}$ the partial multiplicities of λ_i and with $m_{\xi_i,1} \geqslant m_{\xi_i,2} \geqslant \cdots \geqslant m_{\xi_i,d_{\xi_i}}$ be the partial multiplicities of ξ_i. We set $\overline{m}_\lambda := \sum_{i=1}^{s}\sum_{j=1}^{d_i} m_{i,j}$, $\overline{m}_\xi := \sum_{i=1}^{r}\sum_{j=1}^{d_{\xi_i}} m_{\xi_i,j}$ and $\overline{m} = \overline{m_\lambda} + \overline{m_\xi}$. Finally, we define the *spectral matrices* of $F(z)$ to be $\widehat{V} := \check{V}g(\widehat{J})$, $\widehat{W}$, $\widehat{J}$ and $\widetilde{V}$, $\widetilde{W}$, $\widetilde{J}$, as defined in Theorem 2.7.

2.2.2 Counting eigenvalues

In Chapter 3 we will see that estimating the number of eigenvalues s (more precisely, the sum of algebraic multiplicities $\overline{m}$) of a function inside Ω is a fundamental step in most contour algorithms. For the sake of notation, in this section we assume the eigenvalues are simple, so $s = \overline{m}$. Asakura, Sakurai et al. showed that the argument principle in Theorem 1.5 can be used to this goal for a holomorphic function $G(z)$ [Asa+09; Asa+10]. In fact, if $G(z) \in \mathcal{H}(\Omega, \mathbb{C}^{n\times n})$, then

$$s = \frac{1}{2\pi i} \int_{\partial\Omega} \frac{(\det G(z))'}{\det G(z)}\, dz,$$

and Jacobi's formula $(\det G(z))' = \det G(z) \operatorname{tr}(G(z)^{-1} G'(z))$ yields

$$s = \frac{1}{2\pi i} \int_{\partial\Omega} \operatorname{tr}(G(z)^{-1} G'(z))\, dz. \tag{2.12}$$

Equation (2.12) is a bit costly when $n \gg 1$ if estimating the number of eigenvalues is simply a preliminary step for contour algorithms. Even though this chapter mainly focuses on relatively small eigenvalue problems, we believe that a brief review on how to solve this issue is important for the general scope of the thesis.

In general, when $n \gg 1$, an exact count of s is not necessary, hence we may trade some precision for speed. First of all, notice that we can rewrite

$$\operatorname{tr}(G(z)^{-1} G'(z)) = \sum_{j=1}^{n} e_j^T G(z)^{-1} G'(z) e_j, \tag{2.13}$$

where e_j is the jth column of the identity matrix. Hence there are two main ways to approximate (2.12).

- We stochastically estimate $\operatorname{tr}(G(z)^{-1} G'(z))$ with an appropriate random variable. This approach was first proposed in [MFS11] and in many papers onwards, such as [SFT13; Che+17; DPS16].
- We directly approximate $e_j^T G(z)^{-1} G'(z) e_j$ for $j = 1, \dots, n$. In 2004 Guo and Renaut propose to estimate $u^T f(A) v$ through a small number of steps of the Arnoldi iteration for two given vectors u, v [GR04]. Under our settings, we would have $f(A) = A^{-1}$, with $A = G(z)$, $u = e_j$, and $v = G'(z) e_j$.

Here we focus on the first method. The upcoming lemma and proposition are a generalisation of a result first appeared in [Hut90] for symmetric matrices.[1]

1 We thank Vanni Noferini for a private communication on this matter.

Lemma 2.9. *Let $B \in \mathbb{R}^{n\times n}$ and let U be a random variable with zero mean, variance σ^2 and finite fourth moment. Decompose uniquely $B = D + H + K$, where D is diagonal, $H = H^T$ is symmetric with zero diagonal, and $K = -K^T$ is skew-symmetric. If $u \in \mathbb{R}^n$ is a vector of independent samples from U, then*

$$\begin{aligned}\mathbb{E}[u^T Bu] &= \sigma^2 \operatorname{tr}(B),\\ \operatorname{Var}(u^T Bu) &= 2\sigma^4 \|H\|_F^2 + (\mathbb{E}[U^4] - \sigma^4)\|D\|_F^2.\end{aligned}$$

Proof. Denote the (i, j)th element of B with b_{ij}, and respectively h_{ij}, k_{ij} for H and K. We have

$$\mathbb{E}[u^T Bu] = \sum_{j,k=1}^{n} \mathbb{E}[u_j u_k] b_{jk} = \sigma^2 \operatorname{tr}(B).$$

Concerning the variance, it holds $(\mathbb{E}[u^T Bu])^2 = \sigma^4 \sum_{i,j=1}^{n} b_{ii} b_{jj}$. In addition,

$$(u^T Bu)^2 = \sum_{i,j,k,\ell=1}^{n} u_i u_j u_k u_\ell b_{ij} b_{k\ell}. \tag{2.14}$$

Thus, if we take the expected value of (2.14), most of the addenda disappear and we get

$$\mathbb{E}[(u^T Bu)^2] = \sum_{i=1}^{n} \mathbb{E}[U^4] b_{ii}^2 + \sum_{i\neq j} \sigma^4 b_{ii} b_{jj} + \sum_{i\neq j, k\neq \ell} \mathbb{E}[u_i u_j u_k u_\ell] b_{ij} b_{ik}, \tag{2.15}$$

where the first sum corresponds to all indices being equal, and the second to $i = j$ and $k = \ell$. In the third sum two cases return a nonzero contribution: $i = k$, $j = \ell$, and $i = \ell$, $j = k$. Therefore

$$\begin{aligned}\mathbb{E}[(u^T Bu)^2] &= \sum_{i=1}^{n} \mathbb{E}[U^4] b_{ii}^2 + \sum_{i\neq j} \sigma^4 b_{ii} b_{jj} + \sum_{i\neq j} \sigma^4 (b_{ij}^2 + b_{ij} b_{ji})\\ &= \sum_{i=1}^{n} \mathbb{E}[U^4] b_{ii}^2 + \sum_{i\neq j} \sigma^4 b_{ii} b_{jj} + \sum_{i>j} \sigma^4 (b_{ij}^2 + b_{ji}^2 + 2 b_{ij} b_{ji}).\end{aligned}$$

Noticing that $(b_{ij}^2 + b_{ji}^2 + 2b_{ij}b_{ji}) = (b_{ij} + b_{ji})^2$ and that $(b_{ij} + b_{ji}) = 2h_{ij}$ when $i > j$ yields

$$\begin{aligned}\operatorname{Var}(u^T Bu) &= \sum_{i=1}^{n} \mathbb{E}[U^4]b_{ii}^2 + \sum_{i\neq j} \sigma^4 b_{ii}b_{jj} + \frac{\sigma^4}{2}\|2H\|_F^2 - \sigma^4 \sum_{i,j} b_{ii}b_{jj} \\ &= \frac{\sigma^4}{2}\|2H\|_F^2 + (\mathbb{E}[U^4] - \sigma^4)\|D\|_F^2\end{aligned}$$

as desired. □

Proposition 2.10. *Let $B \in \mathbb{C}^{n\times n}$ and let $u \in \mathbb{R}^n$ be as in Lemma 2.9. Decompose uniquely $B = D + H + K$, where D is diagonal, $H = H^*$ is Hermitian with zero diagonal, and $K = -K^*$ is skew-Hermitian. Then*

$$\begin{aligned}\mathbb{E}[u^* Bu] &= \sigma^2 \operatorname{tr}(B), \\ \operatorname{Var}(u^* Bu) &= \frac{\sigma^4}{2}\|2H\|_F^2 + (\mathbb{E}[U^4] - \sigma^4)\|D\|_F^2.\end{aligned}$$

Proof. Write uniquely $B = A + iC$, with $A, C \in \mathbb{R}^{n\times n}$. Since the variance of a complex variable is the sum of the variances of its real and imaginary part, we have

$$\operatorname{Var}(u^* Bu) = \frac{\sigma^4}{2}\|2H_A\|_F^2 + (\mathbb{E}[U^4] - \sigma^4)\|D_A\|_F^2 + \frac{\sigma^4}{2}\|2H_C\|_F^2 + (\mathbb{E}[U^4] - \sigma^4)\|D_C\|_F^2,$$

where $A = D_A + H_A + K_A$ and $C = D_C + H_C + K_C$. But $H = H_A + iH_C$ and similarly for D, hence the result. □

Proposition 2.10 corresponds to [Hut90, Proposition 1] for general matrices. The most widely used random variables are Gaussian and Rademacher vectors. The latter vectors have entries i.i.d. in $\{-1, 1\}$ with equal probabilities and minimise the variance in Proposition 2.10; on the other hand, Gaussian vectors have shown a better numerically convergence to the exact value of the trace [AT11; RA15; WWZ14]. If we define

$$\operatorname{tr}^L(B) := \frac{1}{L}\sum_{j=1}^{L} u_j^* Bu_j \tag{2.16}$$

to be the stochastic estimation of the trace with L random (either Gaussian or Rademacher) vectors, a better measure of its quality instead of the variance is

$$\Pr\left\{\left|\operatorname{tr}(B) - \operatorname{tr}^L(B)\right| \geqslant \varepsilon \operatorname{tr}(B)\right\} \leqslant \delta, \tag{2.17}$$

i.e., the probability that the relative error of the estimator is worse than a given $\varepsilon > 0$ is less than $\delta > 0$. Until recently, the state of the art result by Ubaru, Chen, and Saad for symmetric matrices stated that (2.17) is satisfied with a fixed probability δ when L grows quadratically with respect to n [UCS17]. In the same paper, they showed that the stochastic estimation can be used in pair with the Lanczos method in order to approximate $\operatorname{tr}(f(B))$. More precisely, given the i.i.d. vectors $\{u_j\}_{j=1}^L$, in the first phase one uses m steps of the Lanczos algorithm to approximate $f(B)u_j$, while in the second phase approximate the trace as $\operatorname{tr}^L(f(B))$. Under these settings, (2.17) is satisfied with a linear growth of m and a quadratic one of L with respect to n.

Since $\operatorname{tr}(B)$ is computed exactly in $\mathcal{O}(n^2)$ operations, the fact that (2.17) is satisfied when L grows quadratically with respect to n is not a good property. In 2020, Cortinovis and Kressner proved that the bound is satisfied with a linear growth of L [CK20]. We recall the result for Gaussian vectors and $f(B) = B$, and we refer to the original paper for the other results.

Theorem 2.11 ([CK20, Theorem 5])**.** *Let $B \in \mathbb{R}^{n\times n}$ be a symmetric matrix and let $\operatorname{tr}^L(B)$ the stochastic estimator defined in (2.16) for Gaussian vectors. Then, for every $\varepsilon > 0$,*

$$\Pr\left\{\left|\operatorname{tr}(B) - \operatorname{tr}^L(B)\right| \geqslant \varepsilon \operatorname{tr}(B)\right\} \leqslant \mathrm{e}^{-L\varepsilon/(4\|B\|_F^2+4\varepsilon\|B\|_2)}.$$

Specifically, if $L > 4\varepsilon^{-2}(\|B\|_F^2 + \varepsilon\|B\|_2)\log\left(2\delta^{-1}\right)$, it holds $\Pr\{\left|\operatorname{tr}(B) - \operatorname{tr}^L(B)\right| \geqslant \varepsilon \operatorname{tr}(B)\} \leqslant \delta$.

Remark 2.2. The linear dependence of L on n is not entirely obvious on first sight. Nevertheless, by defining the *stable rank* of B to be $\rho := \|B\|_F^2/\|B\|_2^2$ it holds

$$\frac{4}{\varepsilon^2}(\rho\|B\|_2^2 + \varepsilon\|B\|_2)\log\left(2\delta^{-1}\right) \leqslant \frac{4}{\varepsilon^2}(n\|B\|_2^2 + \varepsilon\|B\|_2)\log\left(2\delta^{-1}\right).$$

Unfortunately, all the previous results show that the quality of the stochastic estimator depends on the off-diagonal elements of $G(z)^{-1}G'(z)$. Since the number of eigenvalues s in Ω depends only on the diagonal elements, even assuming that the same theorems hold for nonsymmetric matrices, it seems there is no way to relate the growth of L with s.

In this section we showed how to estimate the number of eigenvalues when $G(z)$ is a holomorphic function. If instead we have a meromorphic function $F(z)$, then (2.12) returns the difference between the sum of the algebraic multiplicities of the eigenvalues, say s, and the sum of the multiplicities of poles, say r, of the determinant. If r is known or $r \ll s$, then this same strategy allows us to retrieve a good estimate of s nonetheless. However, if $r \approx s$ then (2.12) does not have any practical use for contour algorithms. We refer to Section 3.3 for a deeper look on how to solve this problem.

2.2.3 Backward errors

We are interested in exploring the best ways to compute the backward errors for an approximate eigenpair $(\widehat{\lambda}, \widehat{v})$ of a function $G(z) \in \mathcal{H}(\Omega, \mathbb{C}^{n\times n})$ given in black-box form. First of all, we can define the backward error for $(\widehat{\lambda}, \widehat{v})$ as

$$\eta_G(\widehat{\lambda}, \widehat{v}) := \min\{\varepsilon : \ (G(\widehat{\lambda}) + \Delta G(\widehat{\lambda}))\widehat{v} = 0, \ \|\Delta G\|_\Omega \leqslant \varepsilon \alpha_G\}, \tag{2.18}$$

where $\alpha_G = \|G\|_\Omega$ if we consider the *relative* backward error, and $\alpha_G = 1$ if we consider the *absolute* backward error. This is a similar definition to the one in [GT17, Chapter 2] for matrix-valued functions in split form. The following theorem provides an explicit formula to compute (2.18).

Proposition 2.12. *Let $G(z)\colon \Omega_0 \to \mathbb{C}^{n\times n}$ be a matrix-valued function and $\Omega \subset \Omega_0$. Let $(\widehat{\lambda}, \widehat{v})$ be an approximate eigenpair of $G(z)$ with $\widehat{\lambda} \in \Omega$. Then*

$$\eta_G(\widehat{\lambda}, \widehat{v}) = \frac{\left\|G(\widehat{\lambda})\widehat{v}\right\|_2}{\alpha_G \|\widehat{v}\|_2}, \tag{2.19}$$

where $\eta_G(\widehat{\lambda}, \widehat{v})$ is the backward error defined in (2.18).

Proof. By definition (2.18) $(G(\widehat{\lambda}) + \Delta G(\widehat{\lambda}))\widehat{v} = 0$, so we find that

$$\left\|G(\widehat{\lambda})\widehat{v}\right\|_2 = \left\|\Delta G(\widehat{\lambda})\widehat{v}\right\|_2 \leqslant \left\|\Delta G(\widehat{\lambda})\right\|_2 \|\widehat{v}\|_2 \leqslant \|\Delta G\|_\Omega \|\widehat{v}\|_2 \leqslant \varepsilon \alpha_G \|\widehat{v}\|_2.$$

This already implies

$$\eta_G(\widehat{\lambda}, \widehat{v}) \geqslant \frac{\left\|G(\widehat{\lambda})\widehat{v}\right\|_2}{\alpha_G \|\widehat{v}\|_2}.$$

Finally, consider the perturbation

$$\Delta G(z) = -G(\widehat{\lambda}) \frac{\widehat{v}\widehat{v}^*}{\widehat{v}^*\widehat{v}}.$$

We have that $(G(\widehat{\lambda}) + \Delta G(\widehat{\lambda}))\widehat{v} = 0$, so the first constraint in (2.18) is satisfied. Moreover,

$$\|\Delta G\|_\Omega = \sup_{z \in \Omega} \|\Delta G(z)\|_2 = \frac{\left\|G(\widehat{\lambda})\widehat{v}\right\|_2}{\|\widehat{v}\|_2} = \frac{\left\|G(\widehat{\lambda})\widehat{v}\right\|_2}{\alpha_G \|\widehat{v}\|_2} \alpha_G,$$

thus the proof is completed. □

Remark 2.3. We point out that the explicit formula for the relative backward error (2.19) is not practical in real scenarios, because the denominator contains the norm of $G(z)$ on the continuous set Ω. In numerical experiments we will always substitute $\|G\|_\Omega$ with its best available lower bound. For instance, in this chapter we will often approximate integrals along the contour $\partial\Omega$ with N quadrature points z_j. Hence we will use

$$\max_{1 \leqslant j \leqslant N} \left\|G(z_j)\right\|_2 \lesssim \|G\|_\Omega$$

as an approximation to compute an upper bound on $\eta_G(\widehat{\lambda}, \widehat{v})$ in the relative way.

A natural question is if we can use the relative backward error for meromorphic functions $F(z) \in \mathcal{M}(\Omega_0, \mathbb{C}^{n \times n})$. Unfortunately, definition (2.18) immediately breaks down in that case, because $\|F\|_\Omega = \infty$. We were not able to find a valid alternative to $\eta_F(\widehat{\lambda}, \widehat{v})$ in this situation, unless $F(z)$ is provided in split form, where we can recall the result by Güttel and Tisseur that inspired us

$$\eta_F(\lambda, v) = \frac{\|F(\lambda)v\|_2}{\|v\|_2 \sum_{j=1}^{s} \|A_j\|_F |f_j(\lambda)|},$$

which measures the *relative normwise backward error* for $F(z) = \sum_{j=1}^{s} f_j(z) A_j$ [GT17, Chapter 2]. If we have to limit ourselves to the black-box form of $F(z)$, then the only way around is to consider the absolute backward error.

We point out that $\eta_F(\widehat{\lambda}, \widehat{v})$ is a much better measure on the "goodness" of an eigenpair respect to the residual

$$\frac{\left\|F(\widehat{\lambda})\widehat{v}\right\|_2}{\left\|F(\widehat{\lambda})\right\|_2 \|\widehat{v}\|_2}, \tag{2.20}$$

which is sometimes used in similar works. The following example will show that (2.20) should be avoided at any cost.

Example 2.1. *Let Ω be the unit disk $\mathcal{D}(0,1)$ and $F(z)$ be*

$$F(z) = \begin{bmatrix} z - 0.3 & z^{-1} & 0 \\ 0 & 1 & z^{-1} \\ 0 & 0 & 1 \end{bmatrix}.$$

Assume that a given algorithm returns the eigenpairs

$$(\widehat{\lambda}_1, \widehat{v}_1) = (0.3 + 10^{-10}, e_1 + 10^{-10} \cdot e_2),$$
$$(\widehat{\lambda}_2, \widehat{v}_2) = (10^{-10}, e_1 + 10^{-10} \cdot e_2),$$

where e_j is the jth column of the identity matrix. Note $(\widehat{\lambda}_1, \widehat{v}_1)$ approximates the only eigenpair of $F(z)$, while $(\widehat{\lambda}_2, \widehat{v}_2)$ "approximates" the pole in 0. If we compute the residual with (2.20) we get

$$\frac{\left\|F(\widehat{\lambda}_1)\widehat{v}_1\right\|_2}{\left\|F(\widehat{\lambda}_1)\right\|_2 \|\widehat{v}_1\|_2} = 1.13 \times 10^{-10},$$
$$\frac{\left\|F(\widehat{\lambda}_2)\widehat{v}_2\right\|_2}{\left\|F(\widehat{\lambda}_2)\right\|_2 \|\widehat{v}_2\|_2} = 7 \times 10^{-11},$$

seemingly showing that the two "approximated eigenpairs" are computed with a comparable backward error. However, when we use (2.19) in the absolute way, we get

$$\eta_F(\widehat{\lambda}_1, \widehat{v}_1) = 4.45 \times 10^{-10},$$
$$\eta_F(\widehat{\lambda}_2, \widehat{v}_2) = 0.7,$$

correctly showing that $(\widehat{\lambda}_1, \widehat{v}_1)$ is an approximated eigenpair, while $(\widehat{\lambda}_2, \widehat{v}_2)$ is not.

2.3 THE RIM ALGORITHM

The *Recursive Integral Method* appeared in a work by Huang, Su et al. in 2016 [Hua+16], and was later improved in the following years by the same authors [HSY17; Hua19]. We emphasise that the RIM algorithm was developed to solve *linear* eigenvalue problems of the form $G(z) = A - zB$, but its core principle can be applied more generally. However, the later changes in [HSY17] and [Hua19] exploit the linearity hypothesis and they cannot be generalised further.

Even though it is not the first contour algorithm to have appeared, its simplicity and elegance make it the best choice as the first candidate to be introduced to. Finally, we recall that RIM does not find the eigenpairs (λ, v) in $\Omega \times \mathbb{C}^n \setminus \{0\}$, but only the eigenvalues $\lambda \in \Omega$: more precisely, given $\varepsilon > 0$, it returns $\widetilde{\lambda} \in \Omega$ such that

$$\left|\widetilde{\lambda} - \lambda\right| < \varepsilon \tag{2.21}$$

for $\lambda \in \Lambda(G) \cap \Omega$.

We start by considering the linear problem $(A - \lambda B)v = 0$ and the contour integral

$$\mathcal{I}(p) = \frac{1}{2\pi \mathrm{i}} \int_{\partial\Omega} (A - zB)^{-1} p \, dz, \tag{2.22}$$

where p is a uniformly distributed random vector with unit norm. It immediately follows from Keldysh's theorem 2.4 that

$$\|\mathcal{I}(p)\|_2 \neq 0$$

if at least one eigenvalue $\lambda \in \Omega$ has a left eigenvector w such that $w^* p \neq 0$. If that is the case, we say that Ω is an *admissible region*. This is the elegant observation that already allows us to describe the RIM algorithm:

1. Start with a rectangle region Ω.
2. Approximate the contour integral (2.22) with a quadrature rule of N points $z_1, \ldots, z_N$ and weights $w_1, \ldots, w_N$:

$$\mathcal{I}(p) = \frac{1}{2\pi \mathrm{i}} \int_{\partial\Omega} (A - zB)^{-1} p \, dz \approx \frac{1}{2\pi \mathrm{i}} \sum_{k=1}^{N} w_k (A - z_k B)^{-1} p. \tag{2.23}$$

   - If $\|\mathcal{I}(p)\|_2 = 0$, then exit; otherwise, compute the size of the rectangle Ω. If it is small enough, return its center as the desired eigenvalue λ, else subdivide Ω in S subrectangles Ω_k for $k = 1, \ldots, S$, and call RIM recursively on each Ω_k.

Even though its core is quite simple, there are two main obstacles to overcome. First, computing (2.23) requires the solution of a linear system for each point z_k, which costs $O(n^3)$. In addition, we need to distinguish when $\|\mathcal{I}(p)\|_2$ is 0.

On one hand, reducing the computational cost does not seem possible in the general nonlinear case. On the other hand, when $G(z)$ is linear the authors propose using the Cayley transformation paired with the Arnoldi method [HSY17]. Similarly, having a robust way to determine when $\|\mathcal{I}(p)\|_2$ is zero is not obvious. Theoretically,

Algorithm 2.1: Pseudocode for the RIM algorithm.

```
Input: $G \in \mathcal{H}(\Omega_0, \mathbb{C}^{n\times n})$, $\Omega \subset \Omega_0$, $S$, $\varepsilon$, $\delta$
Output: Set $\Lambda$ of eigenvalues in $\Omega$.
1 Compute $\chi_\Omega$ as in (2.24)
2 if $\chi_\Omega < \delta$ then
3     return {}
4 else if size of $\Omega < \varepsilon$ then
5     return $\Lambda \leftarrow$ {center of $\Omega$};
6 else
7     Subdivide $\Omega$ in $S$ regions $\Omega^{(1)}, \dots, \Omega^{(S)}$
8     return $\Lambda \leftarrow \bigcup_{i=1}^{S} \mathtt{RIM}(G, \Omega^{(i)}, S, \varepsilon, \delta)$
```

it holds $\mathcal{I}(\mathcal{I}(p)) = \mathcal{I}(p)$, however this is no longer true numerically. Hence, in the original article [Hua+16], the authors suggest to use the *indicator*

$$\chi_\Omega := \frac{\|\mathcal{I}(\alpha\mathcal{I}(p))\|_2}{\|\mathcal{I}(p)\|_2}, \tag{2.24}$$

where $\alpha > 0$ is a real constant [Hua+16]. If there are no eigenvalues in Ω, then $\chi_\Omega = o(\alpha)$, while in the opposite scenario $\chi_\Omega \approx \alpha$. The authors set a threshold δ to distinguish the two cases. We summarise the algorithm with this indicator in 2.1.

Notice that (2.24) requires the solution of linear systems with different right-hand side, which means that this indicator is not the optimal choice when one implements RIM in the linear case with the Cayley transformation. Hence the authors later propose

$$\tilde{\chi}_\Omega := \frac{\|\mathcal{I}(p)^{(N)}\|_2}{\|\mathcal{I}(p)^{(2N)}\|_2}, \tag{2.25}$$

where $\mathcal{I}(p)^{(N)}$ is the contour integral approximated with N quadrature points [HSY17]. Assuming that the quadrature rule chosen converges exponentially, it holds

$$\tilde{\chi}_\Omega = \begin{cases} \mathcal{O}(1) & \text{if } \Omega \text{ is admissible,} \\ \mathcal{O}(\mathrm{e}^{-cN}) & \text{otherwise,} \end{cases}$$

therefore (2.25) is another possible indicator, where the region Ω is admissible if $\tilde{\chi}_\Omega = \mathcal{O}(1)$, and $\tilde{\chi}_\Omega = o(1)$ otherwise.

There are other ways in addition to (2.22) to discern whether the target region contains some eigenvalues. Another possibility consists in using the theory described in Section 2.2.2. In fact, (2.12) showed that

$$s = \int_{\partial\Omega} \operatorname{tr}(G(z)G'(z)^{-1})\, dz,$$

under the assumption that the eigenvalues are simple. Hence we can substitute (2.22) and its indicators with

$$\widehat{\chi}_\Omega := \frac{1}{2\pi \mathrm{i}} \int_{\partial\Omega} \operatorname{tr}(G(z)^{-1}G'(z))\, dz. \tag{2.26}$$

There are some trade-offs between using $\mathcal{I}(p)$ with indicators χ_Ω or $\widetilde{\chi}_\Omega$, and $\widehat{\chi}_\Omega$ in (2.26). First, $\widehat{\chi}_\Omega$ does not generalise well to the case when we consider a meromorphic function $F(z) \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$, as opposed to (2.22): if the number of eigenvalues in the region Ω is equal to the number of poles of $\det(F(z))$, then $\widehat{\chi}_\Omega = 0$, while the influence of poles on the other two indicators is not a significant issue. In addition, the computational cost is higher, but still of the same order of magnitude if we trade some memory space. Indeed, for each quadrature point z_j there are n linear systems to solve, thus if we save the LU factorization of $G(z_j)$, then the computation of $\widehat{\chi}_\Omega$ costs $\mathcal{O}(n^3) + n\mathcal{O}(n^2) = \mathcal{O}(n^3)$. On the other hand, $\widehat{\chi}_\Omega$ returns an integer, hence there is no trouble in understanding whether a region is admissible or not. Finally, returning the correct number of eigenvalues in the holomorphic case instead of a single bit of information (Ω is admissible or not) can be exploited in an important way, as we will be showing in the upcoming paragraphs.

2.3.1 A clean-up strategy

The RIM algorithm and its variants have the quality of being very easy to implement and to understand. However, they present some flaws that need to be addressed if one desires to have a robust version. First of all, the original implementation subdivides Ω in $S = 4$ regions. Even though it seems a natural choice, it has a very

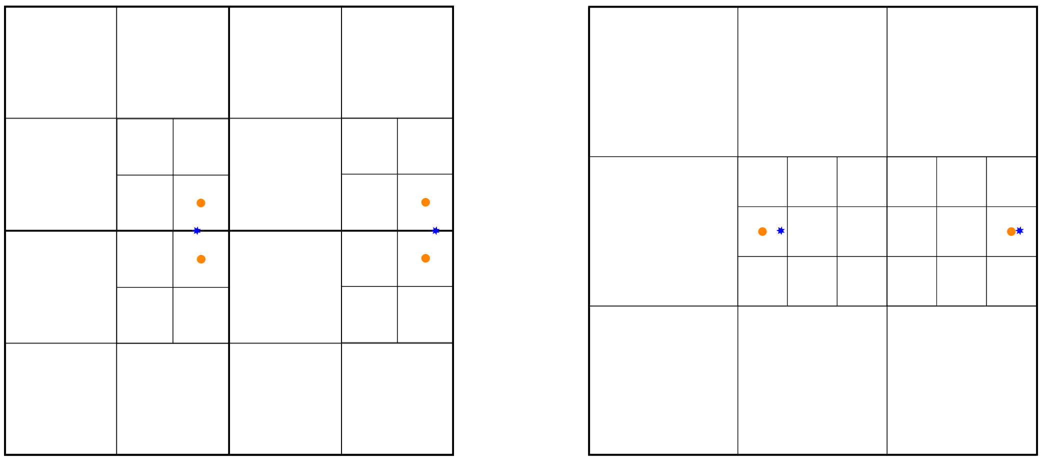


Figure 2.1: On the left, a symmetric region Ω with respect to the real line and $S = 4$: RIM returns two approximations (orange circles) for each real eigenvalue (blue stars); on the right, the same region with $S = 9$: the real eigenvalues are now approximated with a single value.

unpleasant consequence in a common scenario, which we can also witness in many numerical examples in [Hua19]. When Ω is symmetric with respect to the real line, then the first recursion cuts out all the real eigenvalues, which often are the most interesting ones. It follows that for any real eigenvalue λ, RIM returns the conjugate pair $\widetilde{\lambda} \pm ic$, where $\widetilde{\lambda} \approx \lambda$ and $c < \varepsilon$. On one hand, condition (2.21) still holds true, but having a decoupling effect for real (and imaginary eigenvalues when Ω is symmetric with respect to the origin) is not a desirable behaviour. One possible solution is asking the end user to slightly shift Ω diagonally; however, this is not ideal, because any eventual symmetry of $G(z)$ which could help speeding up the algorithm may be lost as well. For instance, if $G(-z) \in \{G(z), -G(z)\}$ or $G(\overline{z}) \in \{G(z), -G(z)\}$, then the algorithm would have to solve only half the linear systems when Ω is symmetric as well. Hence the proposed fix consists in just changing the number of subregions from $S = 4$ to $S = 9$, as depicted in Figure 2.1

The decoupling effect does not happen only on the real line. Preliminary numerical examples 2.2 and 2.3 show that RIM may return a pair of eigenvalues, instead of one, when the correct eigenvalue λ lies near the grid. Unfortunately, we did not find an optimal solution to this issue, but we still propose a cleaning strategy to mitigate it. When the algorithm returns two eigenvalues $\widetilde{\lambda}_1$ and $\widetilde{\lambda}_2$ from adjacent rectangles in the finest grid, we can instead return $\widetilde{\lambda}_3 := (\widetilde{\lambda}_1 + \widetilde{\lambda}_2)/2$.

As the reader may point out, this approach presents some flaws, as well. First, it is not well-defined what the algorithm should do if there are either 3 or 4 adjacent rectangles. Is this phenomenon caused by a single eigenvalue λ near the common corners of the rectangles or by more eigenvalues? In addition, we can not know whether $\widetilde{\lambda}_1$, $\widetilde{\lambda}_2$ are approximations of a single eigenvalue λ near the grid, or of two distinct eigenvalues λ_1 and λ_2. Hence , this heuristic may encounter a false positive and return just an eigenvalue ($\widetilde{\lambda}_3$) instead of two ($\widetilde{\lambda}_1$ and $\widetilde{\lambda}_2$).

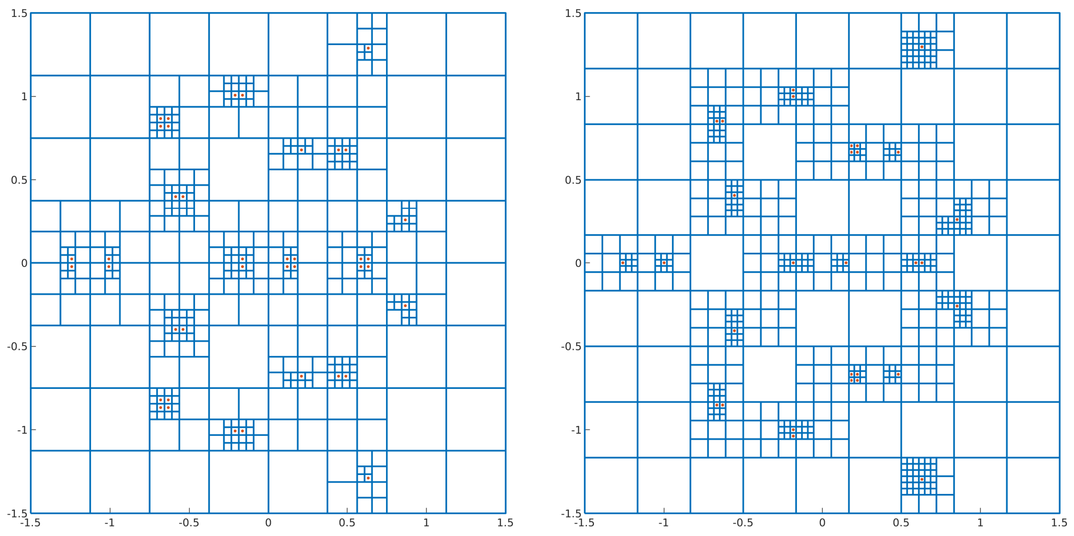
Example 2.2. *We consider a random matrix $A \in \mathbb{C}^{20\times 20}$ and we look for the 19 eigenvalues of the linear problem $G(z) := A - zI$ in the target set $\Omega = [-1.5, 1.5] \times [-1.5, 1.5]$. We set the indicator function to be of $\widehat{\chi}_\Omega$ (2.26) with $\delta = 0.1$, and the threshold $\varepsilon = 10^{-3}$, as in the experiments of [Hua+16]. In Figure 2.2a we plotted the grid of the integrals for $S = 4$ and $\varepsilon = 0.05$: there we can already see the decoupling effect on all the real and some complex eigenvalues; in Figure 2.2b we have set $S = 9$ and we witness that no complex conjugate pairs are returned. Indeed, the algorithm returns 50 approximated eigenvalues when $S = 4$, but "only" 40 when $S = 9$.*

A basic implementation of the cleaning strategy proposed in the previous paragraphs returns 23 and 19 eigenvalues when $S = 4$ and $S = 9$, respectively. This is much better than doing nothing, but is far from perfect: it is clear that if the eigenvalues were laying differently in Ω, then we would have had a different cleaned output. The algorithm takes about 6 seconds to finish, for both choices of the number of subregions; similarly, we do not see a noticeable acceleration when the indicator function is χ_Ω (2.24) or $\widetilde{\chi}_\Omega$ (2.25).

Example 2.3. *In this example we consider the `butterfly` problem*

$$G(z) = z^4 B_4 + z^3 B_3 + z^2 B_2 + z B_1 + B_0, \qquad G(z) \in \mathbb{R}^{64\times 64}[z],$$

from the NLEVP *library [Bet+11]. The matrices B_2 and B_4 are symmetric, while B_1 and B_3 are skew-symmetric. The name `butterfly` comes from the peculiar shape of its spectrum. The target set $\Omega = [-2, 2] \times [-2, 2]$ contains all the 256 eigenvalues of $G(z)$. The parameters are the same as those in Example 2.2, except for the choice of ε, which was set to $\varepsilon_1 = 0.05$ for Figure 2.3 and to $\varepsilon_2 = 0.01$ for the numerical experiment. The algorithm returns 460*



(a) $S = 4$ subregions every recursion. **(b)** $S = 9$ subregions every recursion.

Figure 2.2: The different lattices of contour integrals if one uses a different number of subregions. The choice $S = 4$ return complex conjugate pairs instead of real eigenvalues. The small red points are the approximated eigenvalues.

uncleaned, and 148 *cleaned eigenvalues for* ε_1 *in about* 45 *seconds, while* 572 *and* 270 *for* ε_2 *in* 270 *seconds. Once again, the raw outputs do not give us a good idea on the eigenvalues inside* Ω*. On the other hand, at least when* ε *is small enough, the cleaned output still overestimates the number of eigenvalues in* Ω*, but only by* 14*. Unfortunately, if the threshold* ε *is too large, then neither of the two solutions mirrors the reality of the spectrum of* $G(z)$ *in* Ω*.*

Examples 2.2 and 2.3 highlight two points:

1. the simple clean-up heuristic drastically improves the output, but it is still not perfect;
2. if there are many eigenvalues in Ω, the RIM algorithm considerably slows down.

Concerning the first point, a possible improvement lies in $\widehat{\chi}_\Omega$. Given that $\widehat{\chi}_\Omega$ theoretically returns the number of eigenvalues in Ω, it can decide when it is correct to return all the approximated eigenvalues $\widetilde{\lambda}_i$ or their mean, i.e., $k^{-1}\sum_{i=1}^{k}\widetilde{\lambda}_i$ for $k \in \{2, 3, 4\}$.

A robust, but fast implementation of this strategy is not easy in practice, due to the several edge cases: for example, in Figure 2.4 we plotted a region Ω with 3 eigenvalues (blue stars), one of which lies near the grid. We assume there are 3 levels

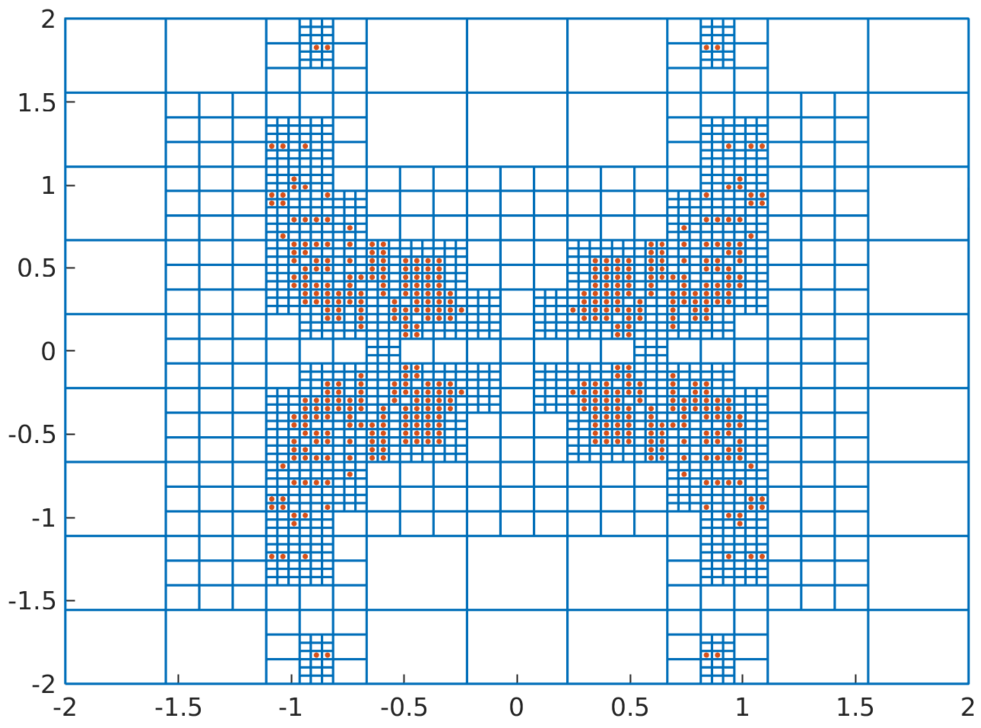


Figure 2.3: RIM integrals to compute the `butterfly` eigenvalues with $\varepsilon = 0.05$.

of recursion and that at the deepest level the algorithm returns the 4 centres (orange circles) of the adjacent squares. At the previous level, the indicator $\widehat{\chi}_\Omega$ realises that there are too many eigenvalues, hence returns the mean value (green circle) for each pair of approximations. At the first level the indicator realises that there are too many eigenvalues (4 instead of 3), but it cannot know which pair of approximated eigenvalues was caused by the decoupling effect, whether the green or the orange circles. The only possibility would lie in computing $\widehat{\chi}_{\Omega'}$, where Ω' is the region enclosed by the blue dashed line, however this would greatly increase the cost of the strategy.

We did not focus more on this clean-up step because its scope is beyond the goal of this thesis, and the timing problem seems an insurmountable obstacle, at least in the nonlinear case where the Cayley transformation strategy cannot be used. Indeed, the algorithms in the upcoming parts of the chapter can retrieve the eigenpairs of Examples 2.2 and 2.3 to a much higher precision in fractions of a second. The difference in timings of orders of magnitude cannot be explained solely by an unoptimised implementation. In fact, RIM needs to compute several integrals when Ω contains

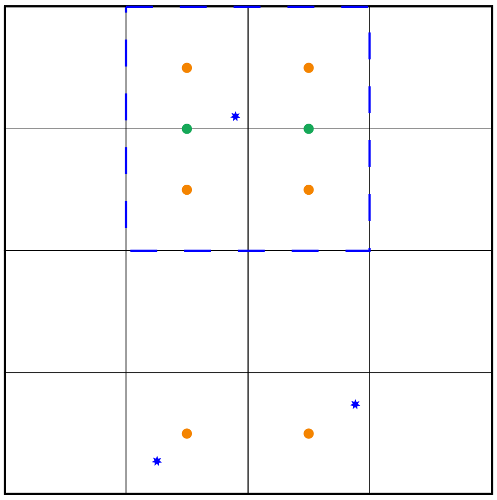


Figure 2.4: An edge scenario which shows that a robust and fast implementation of the cleaning strategy is quite difficult. The blue stars are the eigenvalues, the orange circles are the standard returned approximations, while the green circle the "cleaned" approximations.

many eigenvalues. A cost analysis of the algorithm was still missing in the current literature, hence the next section is dedicated to cover this gap.

2.3.2 The expected computational cost of RIM

As we will see in later sections, the computational cost of the other contour algorithms is predetermined by the size of the matrix function $G(z)$ and by the number of eigenvalues inside Ω. Intuitively, this is not true for RIM. If the eigenvalues are mainly clustered in a small subregion of Ω, then the number of integrals RIM has to compute is smaller than if the eigenvalues were equally spread in Ω. Hence we cannot know a priori the cost, because it will depend on the specific problem. Nevertheless, we can compute the expected value under the hypothesis that there are $s > 0$ eigenvalues and they are uniformly and independently distributed in Ω. In addition, for the sake of simplicity we assume that $\Omega^{(0)} := \Omega = [-1, 1] \times [-1, 1]$.

First of all, the main cost is the approximation of the contour integrals, hence we will compute the expected value of the number of integrals X. Before continuing, we need to set the notation. We denote by $\Omega^{(1)}_{i_1}$ for $i_1 = 1, \dots, S$ the S subregions of Ω and recursively

$$\Omega^{(k)}_{i_1,\dots,i_k} \qquad \text{for } i_k = 1, \dots, S,$$

the S subregions of $\Omega^{(k-1)}_{i_1,\dots,i_{k-1}}$. In addition, we use $N_e(\Omega)$ to denote the number of eigenvalues in Ω.

Let $X^{(k)}$ be the random variable of the number of integrals that RIM computes exactly after the k-th level of recursion. For instance, $X^{(0)} \equiv S$ because we have assumed $s = N_e(\Omega) > 0$. If L is the number of times RIM is recursively called, then

$$\mathbb{E}[X] = 1 + \sum_{k=0}^{L} \mathbb{E}[X^{(k)}], \tag{2.27}$$

where the first addendum represents the first contour integral along $\partial\Omega$. Now, we have

$$\begin{aligned} \mathbb{E}[X^{(1)}] &= \sum_{i_1=1}^{S} S\,\mathcal{P}\{N_e(\Omega^{(1)}_{i_1}) > 0\} \\ &= S \sum_{i_1=1}^{S} \left(1 - \mathcal{P}\{N_e(\Omega^{(1)}_{i_1}) = 0\}\right) = S^2 \left(1 - \mathcal{P}\{N_e(\Omega^{(1)}_{1}) = 0\}\right), \end{aligned} \tag{2.28}$$

where in the last equality we used the hypothesis that the eigenvalues are independently distributed. On the other hand, the uniform distribution tells us that

$$\mathcal{P}\{N_e(\Omega^{(1)}_1) = 0\} = \left(\frac{S-1}{S}\right)^{N_e(\Omega)}. \tag{2.29}$$

In the general case, (2.28) becomes

$$\begin{aligned} \mathbb{E}[X^{(k)}] &= \sum_{i_1,\dots,i_k=1}^{S} S\,\mathcal{P}\{N_e(\Omega^{(k)}_{i_1,\dots,i_k}) > 0\} \\ &= S^k S \left(1 - \mathcal{P}\{N_e(\Omega^{(k)}_{1,\dots,1}) = 0\}\right) = S^{k+1} \left(1 - \left(\frac{S^k - 1}{S^k}\right)^{N_e(\Omega)}\right), \end{aligned} \tag{2.30}$$

where

$$\mathcal{P}\{N_e(\Omega^{(k)}_{1,\dots,1}) = 0\} = \left(\frac{S^k - 1}{S^k}\right)^{N_e(\Omega)},$$

is the generalisation of (2.29) after the k-th level of recursion. Hence, substituting (2.30) in (2.27) yields

$$\mathbb{E}[X] = 1 + \sum_{k=0}^{L} \mathbb{E}[X^{(k)}] = 1 + \sum_{k=0}^{L} S^{k+1}\left(1 - \left(\frac{S^k - 1}{S^k}\right)^{N_e(\Omega)}\right).$$

Finally, the maximum level of recursion L only depends on the number of subregions S and on the threshold ε. When the RIM algorithm exits, it returns the center $\widetilde{\lambda}$ of a subsquare $\Omega^{(L)}_{i_1,\dots,i_L}$ as the approximation of any eigenvalue $\lambda \in \Omega^{(L)}_{i_1,\dots,i_L}$. Thus we have $\left|\widetilde{\lambda} - \lambda\right| < 2\alpha$, where α is half the length of $\Omega^{(L)}_{i_1,\dots,i_L}$'s side. Then it is easy to check that

$$L = \left\lceil -\frac{2\log\varepsilon + \log 2}{\log S} \right\rceil \tag{2.31}$$

is the minimum value that ensures $\left|\widetilde{\lambda} - \lambda\right| < \varepsilon$. We summarise this result in the following proposition.

Proposition 2.13. *Let $G(z) \in \mathcal{H}(\Omega_0, \mathbb{C}^{n\times n})$ and assume that the $N_e(\Omega)$ eigenvalues of $G(z)$ are uniformly and independently distributed in $\Omega := [-1,1] \times [-1,1]$. Let X be the random variable of the number of integrals approximated by the RIM algorithm 2.1. Then*

$$\mathbb{E}[X] = 1 + \sum_{k=0}^{L} S^{k+1}\left(1 - \left(\frac{S^k - 1}{S^k}\right)^{N_e(\Omega)}\right) \leqslant 1 + N_e(\Omega)S(L+1),$$

where L is defined in (2.31).

Proof. We only need to prove the bound. By using Bernoulli's inequality, we have $(1 - S^{-k})^{N_e(\Omega)} \geqslant 1 - N_e(\Omega)S^{-k}$. Therefore

$$S^{k+1}\left(1 - \left(\frac{S^k - 1}{S^k}\right)^{N_e(\Omega)}\right) \leqslant S^{k+1}(1 - 1 + N_e(\Omega)S^{-k}),$$

and then the final bound simply follows. □

2.3.3 Final remarks

We have described and analysed how RIM works for a general holomorphic function $G(z)$ and the differences with the original goal, i.e., linear problems. The simplicity of the theory and of the implementation, which consists in few lines of MATLAB code, makes it an appealing choice for users who are just introduced to contour integral methods. On the other hand, the lack of direct computation of the eigenvectors and the slowdown in the presence of many eigenvalues are noteworthy downsides. Hence the average user should prefer other kind of contour integral methods. The only applications where we could advise to use the RIM algorithm are either as preliminary phase in a multi-step algorithm, or in a scenario where the final user already knows that only few eigenvalues lies in Ω and the required absolute error (2.21) is not too strict.

2.4 BEYN'S ALGORITHM

If RIM and its modifications are based on a *divide et impera* approach, Beyn's algorithm uses directly Keldysh's theorem 2.4 to retrieve all the spectral information in the target set Ω at the same time. In fact, consider the matrix-valued function $G(z) \in \mathcal{H}(\Omega_0, \mathbb{C}^{n\times n})$ and any scalar function $f(z) \in \mathcal{H}(\Omega, \mathbb{C})$. Then, under the Hypotheses 2.1,

$$\begin{aligned}\frac{1}{2\pi \mathrm{i}}\int_{\partial\Omega} f(z)G(z)^{-1}\,dz &= \frac{1}{2\pi \mathrm{i}}\int_{\partial\Omega} f(z)V(zI-J)^{-1}W^*\,dz + \frac{1}{2\pi \mathrm{i}}\int_{\partial\Omega} f(z)R(z)\,dz \\ &= Vf(J)W^*,\end{aligned} \tag{2.32}$$

where in the first equality we used Keldysh's theorem 2.4, while in the second Cauchy's Integral theorem and formula in Theorems 1.1 and 1.3. The goal of Beyn's contour algorithm is to approximate the left-hand side of (2.32) for two (or more) functions $f(z) \in \mathcal{H}(\Omega, \mathbb{C})$ to retrieve the right-hand side through some algebraic manipulations. Further, one usually considers $G(z)^{-1}P$, where $P \in \mathbb{C}^{n\times p}$ is a random matrix with columns of unit norm and $p \leqslant n$ to reduce the computational costs when $\overline{m} < n$.

In 2012, Beyn proposed this strategy to solve the nonlinear eigenvalue problem for a holomorphic function $G(z)$. As written in the introduction, Güttel and Tisseur reported a private conversation with Beyn himself where they realised that the method could work for meromorphic functions as well [GT17, Section 5.5]. Our goal is making this idea precise. We start by recalling the algorithm for holomorphic functions.

2.4.1 The holomorphic case

Consider a function $G \in \mathcal{H}(\Omega_0, \mathbb{C}^{n\times n})$ satisfying the Hypotheses 2.1 and a random matrix $P \in \mathbb{C}^{n\times p}$, with $p \leqslant n$. It follows from (2.32) that

$$\frac{1}{2\pi \mathrm{i}} \int_{\partial\Omega} f(z) G(z)^{-1} P \, dz = V f(J) W^* P. \tag{2.33}$$

The founding bricks of the algorithm are the *(projected) moments*

$$A_k = \frac{1}{2\pi \mathrm{i}} \int_{\partial\Omega} z^k G(z)^{-1} P \, dz = V J^k W^* P, \qquad k = 0, 1, \ldots. \tag{2.34}$$

Remark 2.4. If $\Omega = \mathcal{D}(\gamma, r)$, one often considers the functions $f_k(z) = ((z-\gamma)/r)^k$ for numerical stability. In this way we assure $|f_k(z)| = 1$ on $\partial\Omega$ and we avoid overflows and underflows. The only change in the upcoming analysis happens at the very end, where the eigenvalues computed must be shifted and stretched accordingly.

We divide the analysis in two cases:

- either $\overline{m} \leqslant n$ and $\operatorname{rank}(V) = \overline{m}$, i.e., the number of eigenvalues in Ω is less than n and the matrix of the eigenvectors V is full rank or,
- $\overline{m} > n$ or V is rank-deficient.

In the first scenario, we only consider A_0 and A_1, we write $A_0 = VW^*P$, and assume that the (random) choice of P in (2.33)–(2.34) is such that

$$\operatorname{rank}(W^*P) = \operatorname{rank}(V) = \overline{m}. \tag{2.35}$$

Once more, note that nonlinear problems do not need to have independent (generalised) eigenvectors, therefore $\operatorname{rank}(V) = \overline{m}$ is in fact a necessary assumption. Now, consider the reduced SVD of A_0,

$$A_0 = V_0 \Sigma_0 W_0^*,$$

with $V_0 \in \mathbb{C}^{n\times\overline{m}}$, $\Sigma_0 \in \mathbb{C}^{\overline{m}\times\overline{m}}$ and $W_0 \in \mathbb{C}^{p\times\overline{m}}$. Given that range V_0 = range V, there exists a nonsingular matrix X such that $V = V_0 X$. Therefore $W^*P = X^{-1}\Sigma_0 W_0^*$ and we can write

$$A_1 = VJW^*P = V_0 X J X^{-1} \Sigma_0 W_0^*.$$

It follows that the matrix

$$M := V_0^* A_1 W_0 \Sigma_0^{-1} = XJX^{-1}$$

is equivalent to J and we can therefore retrieve the eigenpairs of the original problem. In fact, if (λ, u) is an eigenpair of M, then λ is an eigenvalue of $G(z)$ and $V_0 u$ is the corresponding eigenvector.

Consider now the case $\overline{m} > n$ or $\operatorname{rank}(V) < \overline{m}$. Then, given $m \in \mathbb{N}$, we can build the following pair of $mn \times mp$ block-Hankel matrices

$$B_0^{[m]} := \begin{bmatrix} A_0 & A_1 & \cdots & A_{m-1} \\ A_1 & A_2 & \cdots & A_m \\ \vdots & \vdots & \iddots & \vdots \\ A_{m-1} & A_m & \cdots & A_{2m-2} \end{bmatrix}, \quad B_1^{[m]} := \begin{bmatrix} A_1 & A_2 & \cdots & A_m \\ A_2 & A_3 & \cdots & A_{m+1} \\ \vdots & \vdots & \iddots & \vdots \\ A_m & A_{m+1} & \cdots & A_{2m-1} \end{bmatrix}, \tag{2.36}$$

which we call the *(projected) Hankel matrices*. The substitution of (2.34) in (2.36) yields the following decompositions:

$$B_0 = OR, \qquad B_1 = OJR \in \mathbb{C}^{mn \times mp}, \tag{2.37}$$

where

$$O = \begin{bmatrix} V \\ VJ \\ \vdots \\ VJ^{m-1} \end{bmatrix} \in \mathbb{C}^{nm \times \overline{m}} \tag{2.38}$$

and

$$R = \begin{bmatrix} W^*P & JW^*P & \cdots & J^{m-1}W^*P \end{bmatrix} \in \mathbb{C}^{\overline{m} \times mp} \tag{2.39}$$

are known as the *observability* matrix and *reachability* matrix in systems theory [AS01]. Furthermore, note that the decompositions (2.37) allows to write the (usually) rectangular pencil as $zB_0 - B_1 = O(zI - J)R$. If the rank of O and R is maximum, i.e., $\operatorname{rank}(O) = \operatorname{rank}(R) = \overline{m}$, then we can retrieve all the spectral information. Therefore we assume that the pair $(m, P) \in \mathbb{N} \times \mathbb{C}^{n \times p}$ is such that

$$\operatorname{rank}(O) = \operatorname{rank}(R) = \overline{m}. \tag{2.40}$$

The natural question is if such condition is attainable, i.e., if there exists m such that $\operatorname{rank}(O) = \overline{m}$. The answer is positive, but we need the following definitions, introduced by Beyn, Effenberger, and Kressner in 2011 [BEK11].

Definition 2.3 (Invariant pair)**.** The pair $(X, \Lambda) \in \mathbb{C}^{n \times k} \times \mathbb{C}^{k \times k}$ is called an *invariant pair* for $G(z)$ if $\mathcal{G}(X, \Lambda) = 0$, where

$$\mathcal{G}(X, \Lambda) = \int_{\partial\Omega} G(z) X (zI - \Lambda)^{-1} \, dz. \tag{2.41}$$

It is easy to see that (V, J) is an invariant pair for $G(z)$. However, Definition 2.3 is not restrictive enough because every pair of the form $(0, J)$ is invariant. Therefore the same authors introduced the concept of *minimal pairs*.

Definition 2.4 (Minimal pair, minimality index)**.** Let $(X, \Lambda) \in \mathbb{C}^{n\times k} \times \mathbb{C}^{k\times k}$ be an invariant pair for $G(z)$. Then (X, Λ) is called a *minimal pair* if there exists an integer m, called *minimality index*, such that

$$\operatorname{rank} \mathcal{V}_m(X, \Lambda) = k,$$

where

$$\mathcal{V}_m(X, \Lambda) = \begin{bmatrix} X \\ X\Lambda \\ \vdots \\ X\Lambda^{m-1} \end{bmatrix}. \tag{2.42}$$

It is now clear that $O = \mathcal{V}_m(V, J)$ and that (2.40) is the condition for (V, J) to be a minimal pair. In general, if we choose m such that $(m-1)p < \overline{m} \leqslant mp$ we may reasonably assume that (2.40) holds true. More precisely, Beyn proved the following bound on the minimality index for (V, J) [Bey12, Lemma 5.1].

Proposition 2.14. *Assume that $G(z) \in \mathcal{H}(\Omega_0, \mathbb{C}^{n\times n})$ satisfies Hypotheses 2.1. Then the minimality index m satisfies*

$$m \leqslant \sum_{j=1}^{s} m_{j,1}.$$

From this point on, the procedure is identical to the first scenario, hence we summarise how to retrieve the eigenpairs in the following theorem. In addition, a pseudocode version is available in Algorithm 2.2, where we assume that all the eigenvalues are simple, i.e., $\overline{m} = s$.

Theorem 2.15 ([Bey12, Theorem 5.2]). *Given the pair $(m, P) \in \mathbb{N} \times \mathbb{C}^{n\times p}$, assume that* $\operatorname{rank} 0 = \operatorname{rank} R = \overline{m}$*, build the block Hankel matrices $B_0^{[m]}$, $B_1^{[m]}$ defined in (2.36). Consider the reduced SVD of $B_0^{[m]} = V_0\Sigma_0 W_0^*$. Then*

$$M := V_0^* B_1^{[m]} W_0 \Sigma_0^{-1} = XJX^{-1}$$

is equivalent to J. Furthermore, if (λ, u) is a right eigenpair of M, then (λ, v) is a right eigenpair of $G(z)$ with $v = V_0^{[1]}u$, where $V_0^{[1]}$ is the upper $n \times \overline{m}$ block of V_0.

Proof. We start by computing an economy-size SVD of $B_0^{[m]}$

$$B_0^{[m]} = V_0 \Sigma_0 W_0^*,$$

where $V_0 \in \mathbb{C}^{mn\times\overline{m}}$, $W_0 \in \mathbb{C}^{mp\times\overline{m}}$ and $\Sigma_0 = \operatorname{diag}(\sigma_1, \dots, \overline{m})$ is invertible due to the rank conditions (2.40). Given that $\operatorname{range}(V_0) = \operatorname{range}(O)$, there exists a nonsingular matrix X such that $O = V_0X$. Therefore we can write $R = X^{-1}\Sigma_0 W_0^*$ and it follows that

$$B_1^{[m]} = OJR = V_0XJX^{-1}\Sigma_0 R,$$

thus the matrix

$$M := V_0^* B_1^{[m]} W_0 \Sigma_0^{-1} = XJX^{-1}$$

is equivalent to J. Hence if (λ, u) is an eigenpair of M, then λ is an eigenvalue of $G(z)$. It is then easy to see that (λ, v) is a right eigenpair of $G(z)$ with $v = V_0^{[1]}u$, where $V_0^{[1]}$ is the upper $n \times \overline{m}$ block of V_0. □

Remark 2.5. Some authors (see, for instance, [Asa+09; Asa+10; GT17]) suggest to use a projection matrix L^* on the left side of $G(z)^{-1}$ to further reduce the size of the problem. However, any choice of $L^* \neq I$ does not allow to compute the right eigenvectors. Hence this is a viable strategy when we are only interested in the eigenvalues.

Algorithm 2.2: Pseudocode for the core of Beyn's algorithm.

```
   Input: $G \in \mathcal{H}(\Omega_0, \mathbb{C}^{n\times n}), \Omega \subset \Omega_0$
   Output: Eigenpairs $(\lambda, v) \in \Omega \times \mathbb{C}^n \backslash \{0\}$.
 1 Choose $(m, P) \in \mathbb{N} \times \mathbb{C}^{n\times p}$ s.t. (2.40) holds
 2 Compute $A_k, k = 0, \dots, 2m-2$
 3 Build $B_0, B_1$
 4 Compute reduced SVD $B_0 = V_0 \Sigma_0 W_0^*$
 5 Compute $M = V_0^* B_1 W_0 \Sigma_0^{-1}$
 6 Extract the eigenpairs $(\lambda, u)$ of $M$
 7 $\Lambda_\Omega \leftarrow \{\}, V_\Omega \leftarrow \{\}$
 8 $V_0^{[1]} = V_0(1:n, 1:N_e(\Omega))$
 9 for $(\lambda_i, u_i) \in \Lambda(M) \times \mathbb{C}^n \backslash \{0\}$ do
10     $\Lambda_\Omega \leftarrow \Lambda \cup \{\lambda_i\}$
11     $V_\Omega \leftarrow V_\Omega \cup \{V_0^{[1]} u_i\}$
12 return $\Lambda_\Omega, V_\Omega$
```

2.4.2 The Loewner interpretation

In 2018 and later in 2020, Brennan, Embree, and Gugercin showed another interpretation of Beyn's algorithm [Bre18; BEG20]. They proved that computing contour integrals as in (2.33) for $G(z) \in \mathcal{H}(\Omega_0, \mathbb{C}^{n\times n})$ can be seen from the point of view of the Loewner framework and thus proposed a variation of the algorithm. In Section 3.4 we will analyse how the approximation of the integral influences the backward error of the eigenvalues, therefore we need to summarise the main differences with the original version.

We consider $G(z) \in \mathcal{H}(\Omega_0, \mathbb{C}^{n\times n})$ satisfying Hypotheses 2.1. From the spectral matrices V, W, J we consider the function

$$H(z) = V(zI - J)^{-1} W^*,$$

which is sometimes known as the *transfer function* in the context of dynamical systems [AS01]. The authors realised that the core of Beyn's algorithm, i.e., the computation of the moment A_k, is equivalent to computing the coefficients of the transfer function formally expanded at infinity. Indeed,

$$\begin{aligned} H(z) &= V(zI - J)^{-1}W^* = z^{-1}V(I - z^{-1}J)^{-1}W^* \\ &= z^{-1}V\left(\sum_{k=0}^{\infty} z^{-k}J^k\right)W^* = \sum_{k=0}^{\infty} VJ^kW^*z^{-(k+1)}. \end{aligned}$$

Note that the previous expansion is always formally correct, however the last two equalities are well-defined as complex-valued functions only if $\|z^{-1}J\| < 1$ for any vector-induced norm. Given this idea, they applied a different strategy: instead of expanding $H(z)$ at ∞, we can expand at specific points $\sigma \in \mathbb{C}\backslash\overline{\Omega}$. In fact, it holds

$$\frac{1}{2\pi \mathrm{i}}\int_{\partial\Omega} \frac{1}{\sigma - z} G(z)^{-1}\, dz = V(\sigma I - J)^{-1}W^* = H(\sigma), \tag{2.43}$$

where as usual we have applied Cauchy's Integral Theorem and Formula (see Theorems 1.1 and 1.3).

Assume now we have at our disposal two sets of r points, $\{\theta_1, \dots, \theta_r\}$, and $\{\sigma_1, \dots, \sigma_r\}$ in $\mathbb{C}\backslash\overline{\Omega}$, which are called the *left interpolation points* and *right interpolation points*, respectively. Similarly, let $\{\ell_1, \dots, \ell_r\}$, and $\{r_1, \dots, r_r\}$ in $\mathbb{C}^n$ be the *left and right directions* associated with the left and right interpolation points, respectively. Finally, we define the so-called *left (interpolation) data* and *right (interpolation) data* to be

$$b_j^* := \ell_j^* H(\theta_j), \qquad c_j := H(\sigma_j) r_j.$$

Thanks to (2.43), we can compute them without explicitly knowing $H(z)$ with

$$b_j^* = \frac{1}{2\pi \mathrm{i}}\int_{\partial\Omega} \frac{1}{\theta_j - z}\ell_j^* G(z)^{-1}\, dz, \qquad c_j = \frac{1}{2\pi \mathrm{i}}\int_{\partial\Omega} \frac{1}{\sigma_j - z} G(z)^{-1} r_j\, dz.$$

It is possible now to retrieve $H(z)$ thanks to the interpolation data. Start by building the Loewner matrices $L, L_s \in \mathbb{C}^{r\times r}$ defined by

$$L_{i,j} = \frac{b_i^* r_j - \ell_i^* c_j}{\theta_i - \sigma_j}, \qquad (L_s)_{i,j} = \frac{\theta_i b_i^* r_j - \sigma_j \ell_i^* c_j}{\theta_i - \sigma_j}$$

or, in explicit form,

$$L = \begin{bmatrix} \frac{b_1^* r_1 - \ell_1^* c_1}{\theta_1 - \sigma_1} & \dots & \frac{b_1^* r_r - \ell_1^* c_r}{\theta_1 - \sigma_r} \\ \vdots & \ddots & \vdots \\ \frac{b_r^* r_1 - \ell_r^* c_1}{\theta_r - \sigma_1} & \dots & \frac{b_r^* r_r - \ell_r^* c_r}{\theta_r - \sigma_r} \end{bmatrix}, \quad L_s = \begin{bmatrix} \frac{\theta_1 b_1^* r_1 - \sigma_1 \ell_1^* c_1}{\theta_1 - \sigma_1} & \dots & \frac{\theta_1 b_1^* r_r - \sigma_r \ell_1^* c_r}{\theta_1 - \sigma_r} \\ \vdots & \ddots & \vdots \\ \frac{\theta_r b_r^* r_1 - \sigma_1 \ell_r^* c_1}{\theta_r - \sigma_1} & \dots & \frac{\theta_r b_r^* r_r - \sigma_r \ell_r^* c_r}{\theta_r - \sigma_r} \end{bmatrix}. \tag{2.44}$$

Furthermore, group the interpolation data in the matrices

$$B = \begin{bmatrix} b_1^* \\ \vdots \\ b_r^* \end{bmatrix} \in \mathbb{C}^{r \times n}, \qquad C = \begin{bmatrix} c_1 & \dots & c_r \end{bmatrix} \in \mathbb{C}^{n \times r}. \tag{2.45}$$

Define now the *generalised observability matrix* $O \in \mathbb{C}^{r \times \overline{m}}$ and *generalised reachability matrix* $R \in \mathbb{C}^{\overline{m} \times r}$, which mirrors (2.38) and (2.39):

$$O = \begin{bmatrix} \ell_1^* (V \theta_1 I_{\overline{m}} - J)^{-1} \\ \vdots \\ \ell_r^* V (\theta_r I_{\overline{m}} - J)^{-1} \end{bmatrix}, \tag{2.46}$$

$$R = \begin{bmatrix} (\theta_1 I_{\overline{m}} - J)^{-1} W^* r_1 & \dots & (\theta_r I_{\overline{m}} - J)^{-1} W^* r_r \end{bmatrix}. \tag{2.47}$$

As one may expect, the observability and the reachability matrix decompose the Loewner and the shifted Loewner matrix.

Theorem 2.16 ([Bre18, Theorem 3.1.2])**.** *Let $L, L_s \in \mathbb{C}^{r \times r}$ be the Loewner matrices defined in* (2.44) *and let O, R be the matrices in (*2.46*–*2.47*). Then*

$$L = -OR, \qquad L_s = -OJR.$$

Theorem 2.16 implies that $L_s - zL = O(zI - J)R$. Hence, in the holomorphic case the condition for the algorithm to work is simply

$$\operatorname{rank}(L) = \operatorname{rank}(O) = \operatorname{rank}(R) = \overline{m}, \tag{2.48}$$

Algorithm 2.3: Pseudocode for the core of the Loewner algorithm.

```
Input: $G \in \mathcal{H}(\Omega_0, \mathbb{C}^{n\times n}), \Omega \subset \Omega_0$
Output: Eigenpairs $(\lambda, v) \in \Omega \times \mathbb{C}^n \backslash \{0\}$.
1  Draw $2r$ points $\{\theta_i\}_{i=1}^r$, $\{\sigma_i\}_{i=1}^r$, $2r$ directions $\{\ell_i\}_{i=1}^r$, $\{r_i\}_{i=1}^r$ s.t. (2.48) holds
2  Compute $H(\sigma_i)$, $H(\theta_i)$ for $i = 1, \dots, r$
3  Build $L, L_s$
4  Compute reduced SVD $B_0 = V_0 \Sigma_0 W_0^*$
5  Compute $M = V_0^* B_1 W_0 \Sigma_0^{-1}$
6  Extract the eigenpairs $(\lambda, u)$ of $M$
7  $\Lambda_\Omega \leftarrow \{\}$, $V_\Omega \leftarrow \{\}$
8  $V_0^{[1]} = V_0(1:n, 1:N_e(\Omega))$
9  for $(\lambda_i, u_i) \in \Lambda(M) \times \mathbb{C}^n \backslash \{0\}$ do
10     $\Lambda_\Omega \leftarrow \Lambda \cup \{\lambda_i\}$
11     $V_\Omega \leftarrow V_\Omega \cup \{V_0^{[1]} u_i\}$
12 return $\Lambda_\Omega, V_\Omega$
```

which mirrors exactly (2.40). We summarise the algorithm in the upcoming theorem, which is the counterpart of Theorem 2.15, and in Algorithm 2.3.

Theorem 2.17. *Given the left points $\{\theta_i\}_{i=1}^r \subset \mathbb{C}\backslash\overline{\Omega}$, the right points $\{\sigma_i\}_{i=1}^r \subset \mathbb{C}\backslash\overline{\Omega}$, and the associated directions $\{\ell_n i\}_{i=1}^r \subset \mathbb{C}^n$, $\{r_i\}_{i=1}^r \subset \mathbb{C}^n$, build the matrices L, L_s in (2.44). Assume that condition (2.48) is satisfied and compute the reduced SVD of $L = V_0 \Sigma_0 W_0^*$. Then*

$$M := V_0^* L_s W_0 \Sigma_0^{-1}$$

contains all the eigenvalues λ_i of $G(z)$ in Ω. Furthermore, if (λ, u) is a right eigenpair of M, then (λ, v) is a right eigenpair of $G(z)$ with $v = V_0^{[1]} u$, where $V_0^{[1]}$ is the upper $n \times \overline{m}$ block of V_0.

Proof. The proof is identical to Theorem 2.15, so we omit it. □

2.4.3 The meromorphic case

We now consider the meromorphic case and look at the differences with the former one. We assume $F(z) \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$ satisfies Hypotheses 2.2 and we use the notation of Table 2.1.

The next result corresponds to Equation (2.32). However, it is not as obvious as its holomorphic counterpart, hence it deserves its own theorem.

Theorem 2.18 (Contour integral for meromorphic matrix-valued functions)**.** *Consider $F \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$ and let $f(z) \in \mathcal{H}(\Omega, \mathbb{C})$. Let $G(z) = g(z)F(z) \in \mathcal{H}(\Omega, \mathbb{C}^{n\times n})$ be the holomorphization of $F(z)$, with $G(z)$ and $g(z) \in \mathbb{C}[z]$ defined in Table 2.1. Then*

$$\frac{1}{2\pi i}\int_{\partial\Omega} f(z)F(z)^{-1}\,dz = \widehat{V}f(\widehat{J})\widehat{W}^* + \widetilde{V}g(\widetilde{J})f(\widetilde{J})\widetilde{W}^* = Vf(J)W^*, \tag{2.49}$$

with

$$V = \begin{bmatrix}\widehat{V} & \widetilde{V}g(\widetilde{J})\end{bmatrix}, \qquad W = \begin{bmatrix}\widehat{W} & \widetilde{W}\end{bmatrix}, \qquad J = \begin{bmatrix}\widehat{J} & \\ & \widetilde{J}\end{bmatrix}.$$

Furthermore, a complete system of Jordan chains in $\widehat{V}$ and $\widehat{W}$ of the eigenvalues λ_k of $F(z)$ satisfy the normalisation conditions (2.2) $\Psi_{\lambda_k}(F,\cdot,\cdot,\cdot)$ for $k = 1,\dots,s$, while a complete system of Jordan chains of the spurious eigenvalues ξ_k in $\widetilde{V}$, $\widetilde{W}$ satisfy the normalisation conditions (2.2) $\Psi_{\xi_k}(G,\cdot,\cdot,\cdot)$ for $k = 1,\dots,r$.

In applications, we are more interested in functions with simple eigenvalues, therefore we state the following corollary, before providing its proof and the one of Theorem 2.18.

Corollary 2.19. *Assume $F(z) \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$ has $\lambda_1,\dots,\lambda_s$ distinct, simple eigenvalues in Ω and $\xi_1,\dots,\xi_r$ distinct poles with the properties summarised in Hypotheses 2.2 and Table 2.1. In addition, let $f(z) \in \mathcal{H}(\Omega, \mathbb{C})$, and $G(z) = g(z)F(z) \in \mathcal{H}(\Omega, \mathbb{C}^{n\times n})$ be the holomorphization of $F(z)$, and $g(z) \in \mathbb{C}[z]$ the corresponding polynomial. Then*

$$\frac{1}{2\pi i}\int_{\partial\Omega} f(z)F(z)^{-1}\,dz = \sum_{k=1}^{s} f(\lambda_k)v_k w_k^* + \widetilde{V}g(J)f(J)\widetilde{W}^*,$$

where v_k, and w_k are the right and left eigenvectors of λ_k satisfying $w_k^* F'(\lambda_k) v_k = 1$, while the complete system of Jordan chains $\widetilde{V}$, $\widetilde{W}$ for the spurious eigenvalues ξ_k satisfy the normalisation conditions (2.2) $\Psi_{\xi_k}(G, \cdot, \cdot, \cdot)$ *for $k = 1, \ldots, r$,*

The main result of Theorem 2.18 is not obvious at first sight. In fact, it seems that it is just a rewording of Equation (2.33). Nevertheless, it is slightly deeper: the decomposition of $F(z)^{-1}$ by Keldysh's theorem for meromorphic functions depends on the polynomial $g(z)$ defined in Lemma 2.6. However, as soon as we compute the contour integral of $F(z)^{-1}$, then the dependence from $g(z)$ on the eigenvalues disappears and remains only for the spurious eigenvalues.

Finally, the only difference between the proof of Theorem 2.18 and the one of Corollary 2.19 is the difficulty that rises with the general normalisation conditions. In a sense, the reader should see Corollary 2.8 as a preparatory lemma for Theorem 2.7, even though it logically follows from it. Therefore we start with the proof of the former result.

Proof of Corollary 2.19. Following the notation of Keldysh's corollary 2.8, we can decompose $F(z)^{-1}$ in Ω as

$$F(z)^{-1} = \sum_{k=1}^{s} \frac{g(z)\breve{v}_k \breve{w}_k^*}{z - \lambda_k} + g(z)\widetilde{V}(zI - \widetilde{J})^{-1}\widetilde{W}^* + g(z)R(z),$$

with $\breve{w}_k^* F'(\lambda_k)\breve{v}_k = g(\lambda_k)^{-1}$ and the complete systems of Jordan chains of the spurious eigenvalues ξ_k satisfying $\Psi_{\xi_k}(G, \cdot, \cdot, \cdot)$ (2.2). By applying Cauchy's integral formula to the contour integral integral we have

$$\frac{1}{2\pi \mathrm{i}} \int_{\partial\Omega} f(z)F(z)^{-1}\, dz = \sum_{k=1}^{s} g(\lambda_k)\breve{v}_k \breve{w}_k^* + \widetilde{V} g(\widetilde{J}) f(\widetilde{J}) \widetilde{W}^*,$$

where the term related to $g(z)R(z)$ disappears because it is holomorphic. The result follows by defining $v_k := g(\lambda_k)\breve{v}_k$ and $w_k := \breve{w}_k$ and noting that

$$w_k^* F'(\lambda_k) v_k = w_k^* F'(\lambda_k)\breve{v}_k g(\lambda_k) = 1. \qquad \square$$

We can now prove Theorem 2.18. We inform the reader that the proof is a bit technical, but not too difficult.

Proof of Theorem 2.18. We follow the steps of the proof of Corollary 2.19 and we apply Keldysh's theorem for meromorphic functions and Cauchy's integral formula in order to get

$$\frac{1}{2\pi \mathrm{i}} \int_{\partial\Omega} f(z)F(z)^{-1} = \check{V}g(J)f(J)\check{W}^{*} + \tilde{V}g(J)f(J)\tilde{W}^{*}, \tag{2.50}$$

where the "reverse hat" matrices correspond to the eigenvalues of $F(z)$, while the "tilde" matrices to its spurious eigenvalues. For each eigenvalue λ_h, the complete system of Jordan chains contained in the in the submatrix $\check{V}_h$ defined in 2.4 satisfies the normalisation conditions $\Psi_{\lambda_h}(G,k,i,j)$ (2.2), i.e.,

$$\Psi_{\lambda_h}(G,k,i,j) = \sum_{\alpha=0}^{k}\sum_{\beta=1}^{m_{h,i}} w_{k-\alpha}^{hj*} \frac{G^{(\alpha+\beta)}(\lambda_h)}{(\alpha+\beta)!} v_{m_{h,i}-\beta}^{hi} = \delta_{ij}\delta_{0k}, \tag{2.51}$$

for $0 \leqslant k \leqslant m_{h,j}-1$, and $1 \leqslant i,j \leqslant d_h$. We will show that if we substitute $\check{V}_{hi}$ with $\hat{V}_{hi} := \check{V}_{hi}g(\hat{J}_{hi})$, then we can substitute $\Psi_{\lambda_h}(G,k,i,j)$ with $\Psi_{\lambda_h}(F,k,i,j)$ and rewrite (2.50) as in the result.

Before starting the core of the proof, we need some preliminary results. First, recall that

$$\check{V}_{hi}g(\hat{J}_{hi}) = \begin{bmatrix} v_0^{hi} & v_1^{hi} & \dots & v_{m_{hi}-1}^{hi} \end{bmatrix} \begin{bmatrix} g(\lambda_h) & g'(\lambda_h) & \cdots & \frac{g^{(m_{hi}-1)}(\lambda_h)}{(m_{hi}-1)!} \\ & g(\lambda_h) & \ddots & \vdots \\ & & \ddots & g'(\lambda_h) \\ & & & g(\lambda_h) \end{bmatrix} =: \hat{V}_{hi},$$

with the $(k+1)$th column of $\hat{V}_{hi}'$ being

$$[\hat{V}_{hi}]_{k+1} = \sum_{\delta=0}^{k} \frac{g^{(\delta)}(\lambda_h)}{\delta!} v_{k-\delta}^{ij} \tag{2.52}$$

for $k = 0, \dots, m_{h,i} - 1$. Now, for the sake of notation, we fix the indices h, i, j, and we drop them, writing $\Psi_{\lambda_h}(G, k, i, j) = \Psi(G, k)$. In addition, recall that if $(v_j)_{j=0}^{m-1}$ is a right Jordan chain for $G(\lambda)$, then by Definition 2.1 we have $v(z) = \sum_{j=0}^{m-1}(z-\lambda)^j v_j$, and $(G(\lambda)v(\lambda))^{(j)} = 0$ for $j = 0, \dots, m-1$. Furthermore, given that $g(\lambda) \neq 0$, it also holds that

$$(F(\lambda)v(\lambda))^{(j)} = 0, \qquad v^{(j)}(\lambda) = j! v_j, \tag{2.53}$$

for $j = 0, \dots, m-1$.

We are now ready to begin. Recall that $G(z) = g(z)F(z)$ and write

$$\Psi(G,k) = \sum_{\alpha=0}^{k}\sum_{\beta=1}^{m} w_{k-\alpha}^* \frac{G^{(\alpha+\beta)}(\lambda)}{(\alpha+\beta)!} v_{m-\beta} = \sum_{\alpha=0}^{k}\sum_{\beta=1}^{m} w_{k-\alpha}^* \sum_{\ell=0}^{\alpha+\beta} \frac{F^{(\ell)}(\lambda)}{\ell!}\frac{g^{(\alpha+\beta-\ell)}(\lambda)}{(\alpha+\beta-\ell)!} v_{m-\beta}$$

It is not too difficult to see that $\sum_{\beta=1}^{m}\sum_{\ell=0}^{\alpha+\beta} = \sum_{\ell=0}^{\alpha}\sum_{\beta=1}^{m} + \sum_{\ell=\alpha+1}^{m+\alpha}\sum_{\beta=\ell-\alpha}^{m}$. Hence we can rewrite the previous equation as

$$\begin{aligned}\Psi(G,k) &= \sum_{\alpha=0}^{k} w_{k-\alpha} \sum_{\ell=0}^{\alpha} \frac{F^{(\ell)}(\lambda)}{\ell!} \sum_{\beta=1}^{m} \frac{g^{(\alpha+\beta-\ell)}(\lambda)}{(\alpha+\beta-\ell)!} v_{m-\beta} \\ &+ \sum_{\alpha=0}^{k} w_{k-\alpha} \sum_{\ell=\alpha+1}^{m+\alpha} \frac{F^{(\ell)}(\lambda)}{\ell!} \sum_{\beta=\ell-\alpha}^{m} \frac{g^{(\alpha+\beta-\ell)}(\lambda)}{(\alpha+\beta-\ell)!} v_{m-\beta}.\end{aligned} \tag{2.54}$$

Let us call the first addendum of (2.54) $\Psi_1(G,k)$, and the second one $\Psi_2(G,k)$. We are going to show that $\Psi_1(G,k) = 0$, while $\Psi_2(G,k) = \Psi(F,k)$. We start from the latter. By substituting the index β with $\delta = \beta + \alpha - \ell$ it holds

$$\Psi_2(G,k) = \sum_{\alpha=0}^{k} w_{k-\alpha} \sum_{\ell=\alpha+1}^{m+\alpha} \frac{F^{(\ell)}(\lambda)}{\ell!} \sum_{\delta=0}^{m+\alpha-\ell} \frac{g^{(\delta)}(\lambda)}{\delta!} v_{m+\alpha-\ell-\delta}.$$

From (2.52) it follows that

$$\widehat{v}_{m+\alpha-\ell} := [\check{V}_{hi} g(\widehat{J}_{hi})]_{m+\alpha-\ell} = \sum_{\delta=0}^{m+\alpha-\ell} \frac{g^{(\delta)}(\lambda)}{\delta!} v_{m+\alpha-\ell-\delta},$$

where the indices h, i in $\widehat{J}_{hi}$ and $\check{V}_{hi}$ were dropped at the beginning of the proof. Hence

$$\Psi_2(G,k) = \sum_{\alpha=0}^{k} w_{k-\alpha}^* \sum_{\ell=\alpha+1}^{m+\alpha} \frac{F^{(\ell)}(\lambda)}{\ell!} \widehat{v}_{m+\alpha-\ell} = \Psi(F,k)$$

by substituting $\beta = \ell - \alpha$. We can now focus on $\Psi_1(G,k)$. We have

$$\Psi_1(G,k) = \sum_{\alpha=0}^{k} w_{k-\alpha} \sum_{\ell=0}^{\alpha} \frac{F^{(\ell)}(\lambda)}{\ell!} \sum_{\beta=1}^{m} \frac{g^{(\alpha+\beta-\ell)}(\lambda)}{(\alpha+\beta-\ell)!} v_{m-\beta}.$$

It is not obvious at first sight, but we can write $\sum_{\alpha=0}^{k}\sum_{\ell=0}^{\alpha} = \sum_{\delta=0}^{k}\sum_{\ell=0}^{k-\delta}$ with $\delta = \alpha - \ell$. Hence

$$\Psi_1(G,k) = \sum_{\delta=0}^{k}\sum_{\ell=0}^{k-\delta} w_{k-\delta-\ell}^* \sum_{\beta=1}^{m} \frac{g^{(\beta+\delta)}(\lambda)}{(\beta+\delta)!} v_{m-\beta}.$$

Now note that the sum indexed by β does not depend on ℓ and furthermore, by (2.53), it holds

$$\sum_{\ell=0}^{k-\delta} w_{k-\delta-\ell}^* \frac{F^{(\ell)}(\lambda)}{\ell!} = \sum_{\ell=0}^{k-\delta} \frac{w(\lambda)^{(k-\delta-\ell)*}}{(k-\delta-\ell)!} \frac{F^{(\ell)}(\lambda)}{\ell!} = (w(\lambda)F(\lambda))^{(k-\delta)},$$

thus $\Phi_1(G,k) = 0$, as claimed. □

We point out that the spurious eigenvalues clearly depend on the polynomial $g(z)$ that define them. However, given a spurious eigenvalue ξ_k, if the size $m_{k,i}$ of any Jordan block $\widetilde{J}_{ki}$ is less than the pole multiplicity c_k, then its contribution will disappear, given that $g(\widetilde{J}_{ki}) = 0$. The following examples and the upcoming proposition clarify this aspect.

Example 2.4. *We set $\Omega = \mathcal{D}(0,6)$, $f(z) \equiv 1$ and consider*

$$F(z) = \begin{bmatrix} \frac{z-1}{(z-2)(z-3)} & \frac{z-4}{z^2} \\ 0 & \frac{z-5}{z-2} \end{bmatrix} \Longrightarrow G(z) = \begin{bmatrix} z^2(z-1) & (z-2)(z-3)(z-4) \\ 0 & z^2(z-3)(z-5) \end{bmatrix},$$

with $g(z) = z^2(z-2)(z-3)$. Then $\lambda_1 = 1$, $\lambda_2 = 5$ are the original eigenvalues of $F(z)$, and $\xi_1 = 0$, $\xi_2 = 3$ are spurious eigenvalues. Following the notation of Theorem 2.18 we can rewrite

$$\widehat{V}\widehat{W}^* = \sum_{j=1}^{2} v_j w_j^*,$$

with $w_j^* F'(\lambda_j) v_j = 1$. Now, the matrix $\widetilde{J}$ consists of the Jordan blocks with eigenvalues ξ_k, for $k = 1, 2, 3$. It is easy to see that ξ_2, ξ_3 are simple eigenvalues, and that the index of ξ_1 is at most 2. Hence $g(\widetilde{J}) = 0$ and the contour integral algorithm will only extract the original eigenvalues of $F(z)$.

Example 2.5. *Consider the matrix-valued function*

$$F(z) = \begin{bmatrix} z-1 & 0 & z^{-1} \\ 0 & z-2 & 0 \\ 0 & 0 & z-3 \end{bmatrix}.$$

from Remark 2.1 on the target set $\Omega = \mathcal{D}(0,4)$. Then $\lambda_j = j$ are the original eigenvalues and $\xi = 0$ is the only spurious eigenvalue. It follows

$$\widetilde{J} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

and hence $g(\widetilde{J}) = \widetilde{J}$ is not zero. Therefore the contour integral (2.49) will contain both the original eigenvectors of $F(z)$ and also one eigenvector associated with the spurious eigenvalue $\xi = 0$.

Lemma 2.20. *Let $g(z) = (z-\xi)^c \in \mathbb{C}[z]$ and let $J \in \mathbb{C}^{\overline{m}\times\overline{m}}$ be a Jordan matrix with d Jordan blocks with eigenvalue ξ. Let $m_1 \geqslant m_2 \geqslant \cdots \geqslant m_d$ be the partial multiplicities that identify J. Then*

$$\operatorname{rank}(g(J)) = \sum_{j=1}^{d} (m_j - c)^+,$$

where $k^+ = \max(0, k)$.

Proof. Denote with $J_i \in \mathbb{C}^{m_i \times m_i}$ each Jordan block of J. Then, by definition of matrix function 1.4, it holds $\operatorname{rank}(g(J_i)) = (m_i - c)^+$, since $g^{(k)}(\xi) = 0$ for $k = 0, \ldots, c-1$. □

Proposition 2.21. *Consider* $F(z) \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$. *Then it holds*

$$\operatorname{rank}(g(\widetilde{J})) = \sum_{i=1}^{r} \sum_{j=1}^{d_{\xi_i}} (m_{\xi_i,j} - c_i)^+, \qquad \operatorname{rank}(\widetilde{V} g(\widetilde{J})) \leqslant \min \left\{ n, \sum_{i=1}^{r} \sum_{j=1}^{d_{\xi_i}} (m_{\xi_i,j} - c_i)^+ \right\},$$

where $k^+ = \max(0, k)$.

Proof. The rank of $g(\widetilde{J})$ is given by the sum of the rank of the matrices $g(\widetilde{J}_i)$ for $i = 1, \ldots, r$, where $\widetilde{J}_i$ is the direct sum of the Jordan blocks with eigenvalue ξ, because $\widetilde{J}$ is block-diagonal. Hence the first part of the result immediately follows from Lemma 2.20. In addition, it holds

$$\widetilde{V} g(\widetilde{J}) = \begin{bmatrix} \widetilde{V}_1 g(\widetilde{J}_1) & \widetilde{V}_2 g(\widetilde{J}_2) & \ldots & \widetilde{V}_r g(\widetilde{J}_r) \end{bmatrix},$$

where

$$\widetilde{V}_i = \begin{bmatrix} \widetilde{V}_{i1} & \ldots \widetilde{V}_{id_i} \end{bmatrix}, \qquad \widetilde{J}_i = \begin{bmatrix} \widetilde{J}_{i1} & & \\ & \ddots & \\ & & \widetilde{J}_{id_i} \end{bmatrix}.$$

Hence, if we fix two indices i and j, we have that the $(k+1)$th column of $\widetilde{V}_{ij} g(\widetilde{J}_{ij})$ is

$$[\widetilde{V}_{ij} g(\widetilde{J}_{ij})]_{k+1} = \sum_{\ell=0}^{k} \frac{g^{(\ell)}(\xi_i)}{\delta!} \widetilde{v}^{ij}_{k-\ell}.$$

for $k = 0, \ldots, m_{\xi_i,j} - 1$. Given that $g^{(\ell)}(\xi_i) = 0$ for $\ell = 0, \ldots, c-1$, it follows that the first c_i columns of $\widetilde{V}_{ij} g(\widetilde{J}_{ij})$ is zero, hence $\operatorname{rank}(\widetilde{V}_{ij} g(\widetilde{J}_{ij})) \leqslant \min\{n, (m_{\xi_i,j} - c_i)^+\}$. Thus

$$\operatorname{rank}(\widetilde{V} g(\widetilde{J})) \leqslant \min \left\{ n, \sum_{i=1}^{r} \sum_{j=1}^{d_i} (m_{\xi_i,j} - c_i)^+ \right\},$$

as claimed in the result. □

We can now show how the algorithm generalises to the meromorphic case. As in the holomorphic case we consider the subcases where it suffices to just consider A_0 and A_1 or when it is necessary to compute more moments. In the first situation, we require the following conditions to be satisfied:

$$\operatorname{rank}(\widehat{V}) = \overline{m}_\lambda, \tag{2.55}$$

$$\operatorname{range}(\widehat{V}) \cap \operatorname{range}(\widetilde{V}g(\widetilde{J})) = \{0\}. \tag{2.56}$$

First, note they are equivalent to (2.35) when the matrix-valued function is holomorphic. In addition, note that if $\overline{m}_\lambda > n$, then (2.55) cannot be satisfied. Similarly, if $\operatorname{rank}(\widehat{V}) + \operatorname{rank}(\widetilde{V}g(\widetilde{J})) > n$, then (2.56) can no longer hold true. These equations formalise the intuition that the algorithm cannot work with only A_0 and A_1 if there are either too many eigenvalues in Ω or too many spurious eigenvalues with algebraic multiplicities larger than their pole multiplicity.

For the sake of notation, we set $P = I$ and we retrace the steps of the holomorphic algorithm. Write

$$A_0 = \begin{bmatrix} \widehat{V} & \widetilde{V}g(\widetilde{J}) \end{bmatrix} \begin{bmatrix} \widehat{W}^* \\ \widetilde{W}^* \end{bmatrix},$$

and consider its SVD

$$A_0 = \begin{bmatrix} \widehat{V}_0 & \widetilde{V}_0 \end{bmatrix} \begin{bmatrix} \widehat{\Sigma} & \\ & \widetilde{\Sigma} \end{bmatrix} \begin{bmatrix} \widehat{W}_0^* \\ \widetilde{W}_0^* \end{bmatrix},$$

where $\widehat{V}_0, \widehat{W}_0 \in \mathbb{C}^{n\times \overline{m}_\lambda}$, $\widetilde{V}_0, \widetilde{W}_0 \in \mathbb{C}^{n\times \overline{m}_\xi}$, and $\widehat{\Sigma} \in \mathbb{C}^{\overline{m}_\lambda \times \overline{m}_\lambda}$, $\widehat{\Sigma} \in \mathbb{C}^{\overline{m}_\xi \times \overline{m}_\xi}$. We can assume that $\operatorname{rank}(A_0) = \overline{m}_\lambda + k$, $k \in \mathbb{N}$ due to (2.55)–(2.56). In addition, (2.56) implies there exist two nonsingular matrices $\widehat{X} \in \mathbb{C}^{\overline{m}_\lambda \times \overline{m}_\lambda}$, $\widetilde{X} \in \mathbb{C}^{\overline{m}_\xi \times \overline{m}_\xi}$ such that $\widehat{V} = \widehat{V}_0\widehat{X}$ and $\widetilde{V}g(\widetilde{J}) = \widetilde{V}_0\widetilde{X}$. It follows that

$$\begin{bmatrix} \widehat{W}^* \\ \widetilde{W}^* \end{bmatrix} = \begin{bmatrix} \widehat{X}^{-1} & \\ & \widetilde{X}^{-1} \end{bmatrix} \begin{bmatrix} \widehat{W}_0^* \\ \widetilde{W}_0^* \end{bmatrix}$$

and we can rewrite A_1 as

$$\begin{aligned} A_1 &= \begin{bmatrix} \widehat{V} & \widetilde{V} \end{bmatrix} \begin{bmatrix} \widehat{J} & \\ & \widetilde{J} \end{bmatrix} \begin{bmatrix} \widehat{W}^* \\ \widetilde{W}^* \end{bmatrix} \\ &= \begin{bmatrix} \widehat{V}_0 & \widetilde{V}_0 \end{bmatrix} \begin{bmatrix} \widehat{X} & \\ & \widetilde{X} \end{bmatrix} \begin{bmatrix} \widehat{J} & \\ & \widetilde{J} \end{bmatrix} \begin{bmatrix} \widehat{X}^{-1} & \\ & \widetilde{X}^{-1} \end{bmatrix} \begin{bmatrix} \widehat{\Sigma} & \\ & \widetilde{\Sigma} \end{bmatrix} \begin{bmatrix} \widehat{W}_0^* \\ \widetilde{W}_0^* \end{bmatrix}. \end{aligned}$$

Hence the matrix

$$M_1 = \begin{bmatrix} \widehat{V}_0^* \\ \widetilde{V}_0^* \end{bmatrix} A_1 \begin{bmatrix} \widehat{W}_0\widehat{\Sigma}^{-1} & \widetilde{W}_0\widetilde{\Sigma}^{\dagger} \end{bmatrix} = \begin{bmatrix} X\widehat{J}X^{-1} & \\ & X\widetilde{J}X^{-1} \end{bmatrix}$$

allows us to retrieve the original eigenvalues and eigenvectors of $F(z)$, as we did in the holomorphic case, plus a subset of the spurious eigenvalues. We underline the fact that the algorithm does not really change: one always computes a reduced SVD of size $\overline{m}_\lambda + k$: the previous analysis only shows that it is still able to retrieve the spectral information of $F(z)$.

As we can generalise the holomorphic algorithm to the case where we need to compute A_k for $k > 1$, the same is true in the meromorphic case. The main difficulty lies in finding m such that $B_0^{[m]}$ and $B_1^{[m]}$ contain the spectral information we are interested in.

Definition 2.5 (Meromorphic minimality index)**.** Consider $F(z) \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$. Decompose $B_0^{[m]}$ in

$$B_0^{[m]} = \begin{bmatrix} \widehat{V} & \widetilde{V}g(\widetilde{J}) \\ \widehat{V}\widehat{J} & \widetilde{V}g(\widetilde{J})\widetilde{J} \\ \vdots & \vdots \\ \widehat{V}\widehat{J}^{m-1} & \widetilde{V}g(\widetilde{J})\widetilde{J}^{m-1} \end{bmatrix} \begin{bmatrix} \widehat{W}^* & \widehat{J}\widehat{W}^* & \dots & \widehat{J}^{m-1}\widehat{W}^* \\ \widetilde{W}^* & \widetilde{J}\widetilde{W}^* & \dots & \widetilde{J}^{m-1}\widetilde{W}^* \end{bmatrix} = \begin{bmatrix} \widehat{O} & \widetilde{O} \end{bmatrix} \begin{bmatrix} \widehat{R}^* \\ \widetilde{R}^* \end{bmatrix},$$

where $\widehat{O}, \widehat{R} \in \mathbb{C}^{nm \times \overline{m}_\lambda}$ and $\widetilde{O}, \widetilde{R} \in \mathbb{C}^{nm \times \overline{m}_\xi}$. We say that m is the *meromorphic minimality index* if it is the smallest integer such that

$$\operatorname{rank}(\widehat{O}) = \overline{m}_\lambda, \tag{2.57}$$

$$\operatorname{range}(\widehat{O}) \cap \operatorname{range}(\widetilde{O}) = \{0\}. \tag{2.58}$$

It is clear the meromorphic minimality index is bounded by $\sum_{j=1}^{s} m_{j,1} + \sum_{j=1}^{r} m_{\xi_j,1}$ by applying Proposition 2.14 to $G(z) = g(z)F(z)$. However, since we are only interested in the original eigenvalues and not in the spurious ones, this bound is quite loose. Consider, for instance,

$$F(z) = \begin{bmatrix} z-1 & 1 \\ 0 & z^{-\ell} \end{bmatrix} \tag{2.59}$$

with $\ell \in \mathbb{N}$ and target set $\Omega = \mathcal{D}(0,2)$. On one hand, Proposition 2.14 applied to $G(z) = z^\ell F(z)$ tells us that the minimality index is bounded by $\ell + 1$; on the other, it is easy to see that the conditions (2.55-2.56) are satisfied because the spurious eigenvalue ξ is filtered out by the contour integral. Therefore it is natural to ask ourselves if there exists a sharper bound. The upcoming theorem, which mirrors Proposition 2.14, gives a positive answer to this question.

Proposition 2.22. *Let $F(z) \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$ satisfying the Hypotheses 2.2. Then the meromorphic minimality index m for F is bounded from above by*

$$m \leqslant \sum_{i=1}^{s} m_{i,1} + \sum_{i=1}^{r} (m_{\xi_i,1} - c_i)^+.$$

Proof. We follow the steps of [Bey12, Lemma 5.1]. Let $M = \sum_{i=1}^{s} m_{i,1} + \sum_{i=1}^{r}(m_{\xi_i,1} - c_i)^+$ and assume that $VJ^jx = 0$, for some $x \in \mathbb{C}^n$ and $j = 0, \dots M-1$, with $V = [\widehat{V}, \widetilde{V}g(\widetilde{J})]$. For the sake of notation denote $\lambda_{s+i} = \xi_i$ for $i = 1, \dots r$ and

$$\widetilde{m}_{i,1} = \begin{cases} m_{i,1} & i = 1, \dots s, \\ m_{\xi_{i-s},1} - c_{i-s} & i = s+1, \dots s+r. \end{cases}$$

For any $1 \leqslant i \leqslant r+s$ and $0 \leqslant \beta \leqslant \widetilde{m}_{i,1}$, define the polynomial

$$P_{i,\beta}(z) = (z-\lambda_i)^\beta \prod_{j=1, j\neq i}^{r+s} (z-\lambda_i)^{\widetilde{m}_{i,1}}.$$

By our assumption, it follows that $VP_{i,\beta}(J)x = 0$. We have to partition $V = [\widehat{V}\ \widetilde{V}\widetilde{J}]$, $J = \operatorname{diag}(\widehat{J},\ \widetilde{J})$, and $x^T = [\widehat{x}^T\ \widetilde{x}^T]$ in the usual way:

$$\begin{aligned}
\widehat{x} &= \begin{bmatrix} \widehat{x}_1 & \dots \widehat{x}_s \end{bmatrix}, & \widehat{x}_i &= \begin{bmatrix} \widehat{x}_{i1} & \dots \widehat{x}_{id_i} \end{bmatrix}, & \widehat{x}_{ij} &= \begin{bmatrix} \widehat{x}^{ij}_0 & \dots \widehat{x}^{ij}_{m_{i,j}-1} \end{bmatrix}, \\
\widehat{V} &= \begin{bmatrix} \widehat{V}_1 & \dots \widehat{V}_s \end{bmatrix}, & \widehat{V}_i &= \begin{bmatrix} \widehat{V}_{i1} & \dots \widehat{V}_{id_i} \end{bmatrix}, & \widehat{V}_{ij} &= \begin{bmatrix} \widehat{v}^{ij}_0 & \dots \widehat{v}^{ij}_{m_{i,j}-1} \end{bmatrix}, \\
\widehat{J} &= \begin{bmatrix} \widehat{J}_1 & & \\ & \ddots & \\ & & \widehat{J}_s \end{bmatrix}, & \widehat{J}_i &= \begin{bmatrix} \widehat{J}_{i1} & & \\ & \ddots & \\ & & \widehat{J}_{id_i} \end{bmatrix},
\end{aligned}$$

where $\widehat{J}_{ij}$ is defined in (2.6) with the analogous for the tilde matrices $\widetilde{V}$, $\widetilde{J}$, and vector $\widetilde{x}$. Once more we denote with $J_i := \widehat{J}_i$ for $1 \leqslant i \leqslant s$ and with $J_i := \widetilde{J}_{i-s}$ for $s+1 \leqslant i \leqslant r+s$. Now we fix an index $i \leqslant s$. We recall that for $j \neq i$ it holds $(\widehat{J}_j - \lambda_j I)^{\widetilde{m}_{j1}} = 0$ if $j \leqslant s$ and that $\widetilde{V}_{j-s} g(\widetilde{J}_{j-s})(\widetilde{J}_{j-s} - \xi_{j-s} I)^{m_{\xi_{i,1}} - c_i}$ if $s < j \leqslant r$. Thus we can rewrite $0 = VP_{i,\beta}(J)x$ as

$$0 = \sum_{\ell=1}^{d_i} V_{i\ell} \prod_{\substack{j\neq i \\ \widetilde{m}_{j,\ell}-1 \geqslant \beta}}^{r+s} (J_{i\ell} - \lambda_j I)^{\widetilde{m}_{j,1}} (J_{i\ell} - \lambda_i I)^\beta x_{i\ell}. \tag{2.60}$$

Now we are going to prove by induction on $\beta = m_{i,1} - 1, \dots, 0$ that

$$\widehat{x}^{i\ell}_\delta = 0, \qquad \text{for } \beta \leqslant \delta \leqslant m_{i,\ell} - 1. \tag{2.61}$$

In the base case $\beta = m_{i,1} - 1$, all the Jordan blocks of sizes strictly less than $m_{i,1}$ disappear, hence (2.60) becomes

$$0 = \sum_{j\neq i}^{r+s} (\lambda_j - \lambda_i)^{\widetilde{m}_{j,1}} \sum_{\substack{\ell=1 \\ m_{i,\ell} = m_{i,1}}}^{d_i} v^{i\ell}_0 x^{i\ell}_{m_{i,\ell}-1},$$

and (2.61) holds true because the vectors $v_0^{i\ell}$ are linearly independent, as per Definition 2.1. Now it is time for the induction step. By putting $\beta - 1$ in (2.60) and using the induction hypothesis, it follows that

$$0 = \sum_{j \neq i}^{r+s} (\lambda_j - \lambda_i)^{\widetilde{m}_{j,1}} \sum_{\substack{\ell=1 \\ m_{i,\ell} \geqslant \beta}}^{d_i} v_0^{i\ell} x_{\beta-1}^{i\ell},$$

therefore (2.61) is always true. Now fix an index $1 \leqslant i \leqslant r$. Similarly to the previous case we can rewrite $0 = VP_{i,\beta}(J)$ as

$$0 = \sum_{\ell=1}^{d_{\xi_i}} \widetilde{V}_{(i-s)\ell} g(J_{i\ell}) \prod_{\substack{j \neq i \\ \widetilde{m}_{j,\ell}-1 \geqslant \beta}}^{r+s} (J_{i\ell} - \lambda_j I)^{\widetilde{m}_{j,1}} (J_{i\ell} - \lambda_i I)^{\beta} x_{i\ell}, \tag{2.62}$$

and we are going to show by induction on $\beta = m_{\xi_i,1} - c_i - 1, \ldots, 0$ that

$$x_\delta^{i\ell} = 0, \qquad \text{for } \beta + c_i \leqslant \delta \leqslant m_{\xi_i,\ell} - 1. \tag{2.63}$$

Note that (2.63) and (2.63) together prove the result. In fact, they show that the first $\overline{m}_\lambda$ columns are independent, that the rank of the following $\overline{m}_\xi$ is

$$\sum_{i=1}^{r} \sum_{j=1}^{d_{\xi_i}} (m_{\xi_i,j} - c_i)^+,$$

and that they also are independent from each other.

For the base case of the induction $\beta = m_{\xi_i,1} - c_i - 1$, note that all the Jordan blocks $J_{i\ell}$ of size strictly less than $m_{\xi_i,1}$ disappear, due to the contribution of

$$g(J_{i\ell}) = (J_{i\ell} - \lambda_i)^{c_i} \prod_{\substack{j=s+1 \\ j \neq i}}^{r+s} (J_{i\ell} - \lambda_j)^{c_j}.$$

Therefore (2.62) becomes

$$0 = \sum_{j \neq i}^{r+s} (\lambda_j - \lambda_i)^{\widetilde{m}_{j,1}} \sum_{\substack{\ell=1 \\ m_{i,\ell} = m_{i,1}}}^{d_i} v_0^{i\ell} x_{m_{i,\ell}-1}^{i\ell},$$

Algorithm 2.4: Pseudocode for the core of Beyn's algorithm in the meromorphic case.

```
   Input: $F \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n}), \Omega \subset \Omega_0$
   Output: Eigenpairs $(\lambda, v) \in \Omega \times \mathbb{C}^n \backslash \{0\}$.
1  Choose $(m, P) \in \mathbb{N} \times \mathbb{C}^{n\times p}$ s.t. (2.57–2.58) hold
2  Compute $A_k$, $k = 0, \dots, 2m-2$
3  Build $B_0, B_1$
4  Compute reduced SVD $B_0 = V_0 \Sigma_0 W_0^*$
5  Compute $M = V_0^* B_1 W_0 \Sigma_0^{-1}$
6  Extract the eigenpairs $(\lambda, u)$ of $M$
7  $\Lambda_\Omega \leftarrow \{\}$, $V_\Omega \leftarrow \{\}$
8  $V_0^{[1]} = V_0(1:n, 1:N_e(\Omega))$
9  for $(\lambda_i, u_i) \in \Lambda(M) \times \mathbb{C}^n$ do
10     if $\|F(\lambda)\| < \infty$ then
11         $\Lambda_\Omega \leftarrow \Lambda \cup \{\lambda_i\}$
12         $V_\Omega \leftarrow V_\Omega \cup \{V_0^{[1]} u_i\}$
13 return $\Lambda_\Omega, V_\Omega$
```

and the base case holds once more for the linear independence of $v_0^{i\ell}$. Finally, the induction step is identical: we insert $\beta - 1$ in (2.62) and get

$$0 = \sum_{\ell=1}^{d_{\xi_i}} \widetilde{V}_{(i-s)\ell}\, g(J_{i\ell}) \prod_{\substack{j\neq i \\ \widetilde{m}_{j,\ell}\geqslant\beta}}^{r+s} (J_{i\ell} - \lambda_j I)^{\widetilde{m}_{j,1}} (J_{i\ell} - \lambda_i I)^{\beta-1} x_{i\ell}.$$

By using the induction hypothesis (2.63) it holds that $g(J_{i\ell})(J_{i\ell} - \lambda_i)^{\beta-1} x_{i\ell} = x_{\beta-1}^{i\ell} e_1$, so that

$$0 = \sum_{j\neq i}^{r+s} (\lambda_j - \lambda_i)^{\widetilde{m}_{j,1}} \sum_{\substack{\ell=1 \\ m_{i,\ell}\geqslant\beta}}^{d_i} v_0^{i\ell} x_{\beta-1}^{i\ell},$$

therefore (2.63) holds true and we have shown that all the components of x are zero, except at most the ones corresponding to $\widetilde{x}_\delta^{i\ell}$ for $1 \leqslant i \leqslant r$, for $1 \leqslant \ell \leqslant d_{\xi_i}$, and for $0 \leqslant \delta \leqslant \min\{m_{\xi_i,\ell}, c_i - 1\}$. □

In Algorithm 2.4 we summarised the core of the procedure in the meromorphic case. The only difference with Algorithm 2.2 arises in the final retrieval of the eigenpairs. Given that we may extract spurious eigenvalues ξ from the matrix M, we need to make sure that we do not return them to the final user. Theoretically, we should check that all the entries of $F(\xi)$ are finite; in practice we discard ξ when by checking $\|F(\xi)\|$ and discarding ξ when $\max_{1\leqslant i,j\leqslant n} \left|[F(\xi)]_{ij}\right| \gg 1$. As the reader may point out, at the moment we left some details out. For instance, we did not focus on how well to approximate the moments A_k or, more importantly, on how to choose m and P. It is clear that this choice heavily depends on the number of eigenvalues of $F(z)$, which we cannot assume we know a priori. We were not able to find a sufficient number of theoretical results to justify all our choices on the algorithm's parameters. Hence, we preferred to draw a line by ending the theoretical chapter here and create a new chapter, where we explore the topics mentioned above from a numerical point of view.

3 PRACTICAL CONSIDERATIONS ON CONTOUR INTEGRAL METHODS

3.1 INTRODUCTION

In the previous chapter we focused our attention on contour integral methods to solve nonlinear eigenvalue problems. Our goal was showing how Beyn's algorithm changes when the input is a meromorphic function $F(z) \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$. The main goal of this chapter is to describe a practical implementation of a contour algorithm that will allow almost any end-user to solve small and mid-sized nonlinear eigenvalue problems without a deep knowledge of the numerical linear algebra that lies behind it. In a sense, the philosophy behind it is the same of `eig` or `\` in MATLAB: anybody can use them without knowing the precise algorithms they are based on.

Most of the solutions we present here are not based on theoretical results, but on heuristics that we found through many numerical experiments. One of the initial problems we had to face was the absence of a large set of examples which we could calibrate our algorithm with. The NLEVP library is a collection of problems that researchers can use to benchmark their software [Bet+11]. Unfortunately, the 3.0 version was dated back to 2013 and was mainly focused on quadratic and polynomial eigenvalue problems. Hence we realised it was time to update it with newer problems and newer features. Its 4.0 incarnation is the topic of Section 3.2.

Furthermore, we saw that the algorithms of the previous chapter require the choice of multiple parameters in order to retrieve the eigenpairs: even excluding the quadrature rule to use and the number of quadrature points, Beyn's algorithm needs to set the number m of moments to compute and the number p of columns of the probing matrix $P \in \mathbb{C}^{n\times p}$; on the other hand, the Loewner interpretation has to set not only the number r of interpolation points and directions, but also the interpolation points σ_i and θ_i themselves. In Section 3.3 we show how we automatically set the many parameters Beyn's algorithm needs without having the user to worry about them. In Section 3.4 we analyse the influence of the quadrature approximation. There we generalise the results of Beyn to the meromorphic case and we see how the presence of poles changes the convergence to the exact value of the integrals. In Section 3.5 we cover the refinement strategies. Given the broad scope we have prefixed to ourselves, hoping that we will always be able to retrieve the eigenpairs with satisfactory precision is too optimistic. Hence we propose different refinement strategies to avoid restarting the algorithm from scratch. Finally, we postpone most of the numerical experiments to Section 4.4 of Chapter 4, where we compare this implementation to the algorithms described there.

3.2 THE NLEVP 4.X LIBRARY

In numerical analysis having access to a set of problems is a fundamental tool for research. It helps tuning and benchmarking your own algorithms against a wide range of situations. In addition, it allows researchers of different groups to test their work on the same field.

The history of numerical linear algebra is filled with numerous problem collections. For example, the MATLAB function `gallery` contains more than 50 examples of matrices with interesting properties; similarly, the SuiteSparse Matrix Collection (formerly known as the The University of Florida Sparse Matrix Collection) contains more than 2800 sparse matrices [Dav97; DH11]. The NLEVP library was born on the wave of the growing interest for nonlinear eigenvalue problems in 2008 and contained 26 problems; two years later version 2.0 was released with the addition of another 20 problems, and finally Betcke, Higham, Mehrmann, Schröder, and Tisseur published 3.0 in 2011 for a total of 52 problems [Bet+11]. It was immediately widely adopted and many problems, like `gun`, started to appear in several papers interested in either eigenvalues or rational approximations. One of the main perk of the toolbox is the easiness of installation and use under MATLAB. Given a matrix-valued function named `problem` of the form

$$F(z) = \sum_{j=1}^{s} f_j(z) A_j$$

then the command

```
[coeffs, fun] = nlevp('problem')
```

would return a cell `coeffs` containing the matrices A_j, and a function handle array `fun` with the scalar functions f_j. In addition, all the problems were described by their properties, such as `real`, `sparse`, or `symmetric`.

As hinted in the introduction, we realised it was time to update the library, in order to add both new features and new nonpolynomial problems. In fact, we were not the only one having this idea: in 2018 Jarlebring, Bennedich et al. developed NEP-PACK, a JULIA package for nonlinear eigenvalue problems [Jar+18]. In 2019 we finally released version 4.0, which contained 22 new problems. We also introduced three new identifiers, `tridiagonal`, `banded`, and `low-rank`, together with a third output: the command

```
[coeffs, fun, F] = nlevp('problem')
```

now returns a matrix-valued function handle `F` for $F(z)$. This substitutes the original and slower inline call

```
F = @(z) nlevp('eval', name, z).
```

Finally, a fourth output, `xcoeffs`, is available for all the problems with the `low-rank` identifier. It contains a $2 \times s$ cell such that if the kth matrix coefficient $A_k = B_k C_k$ for some rectangular matrices B_k, C_k, then `xcoeffs{1,k}` $= B_k$ and `xcoeffs{2,k}` $= C_k$.

Since version 4.0, the library is hosted on MATLAB File Exchange and on the GITHUB repository https://github.com/ftisseur/nlevp. This choice allows an easier way for us to update and add other problems, but also for the researchers to download and use it. The latest public version is the 4.1, which consists of 80 problems. In Table 3.1 we have listed the ones we added since 3.0.

Table 3.1: New problems in NLEVP version 4.0 and 4.1.

Quadratic	bcc_traffic	circular_piston	damped_gyro
	deformed_consensus	disk_brake100	disk_brake4669
	elastic_deform	utrecht1331	
Rational	photonic_crystal	railtrack_rep	railtrack2_rep
Nonlinear	bent_beam	bucking_plate	canyon_particle
	clamped_beam_1d	distributed_delay1	nep1
	nep2	nep3	neuron_dde
	pdde_symmetric	pillbox_cavity	pillbox_small
	sandwich_beam	schrodinger_abc	square_root
	time_delay2	time_delay3	

3.3 THE CHOICE OF THE PARAMETERS

In the previous chapter we generalised Beyn's algorithm to meromorphic functions $F(z) \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$. Furthermore, we saw that the same can be easily done for its Loewner interpretation. It is clear that these two approaches share many similarities. The former opened the paths of using contour integrals to solve eigenvalue problems, while the latter shed some light on the link between contour integrals and the Loewner framework.

Here we mainly focus on the original interpretation. We consider $F(z) \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$ satisfying the Hypotheses 2.2. In addition, we assume:

1. the eigenvectors in $\widehat{V}$ are linearly independent when $\overline{m}_\lambda \leqslant n$, i.e., when the number of eigenvalues in Ω is less than n;

2. for each spurious eigenvalue ξ_i, $c_i \geqslant m_{\xi_i,1}$, i.e., its pole multiplicity is greater than the size of its largest Jordan block as spurious eigenvalue of $F(z)$.

The first assumption is quite natural and it implies that the algorithm works with the two moments A_0 and A_1 whenever the number of eigenvalues in Ω is smaller than n. The second one implies that $\widetilde{V}g(\widetilde{J}) = 0$, therefore (2.58), i.e.,

$$\operatorname{range}(\widehat{O}) \cap \operatorname{range}(\widetilde{O}) = \{0\},$$

is always satisfied. In a sense, this assumption is a natural extension to working with simple eigenvalues in the holomorphic case. Given these premises, the algorithm necessitates (2.57) to be satisfied, i.e.,

$$\operatorname{rank}(\hat{O}) = \operatorname{rank}(\hat{R}) = \overline{m}_\lambda.$$

A necessary condition for this to be true is that $\min(nm, mp) > \overline{m}_\lambda$. Given that it would not make sense for $P \in \mathbb{C}^{n\times p}$ to be a "fat" matrix, i.e., $p > n$, the condition becomes

$$mp > \overline{m}_\lambda. \tag{3.1}$$

It is therefore clear that estimating the number of eigenvalues in Ω is a fundamental preliminary step of the contour algorithms. For sake of the notation we assume that all the eigenvalues are simple, so that $\overline{m}_\lambda = s$. In Section 2.2.2 we have seen that, for a holomorphic function $G(z) \in \mathcal{H}(\Omega_0, \mathbb{C}^{n\times n})$,

$$s = \frac{1}{2\pi \mathrm{i}} \int_{\partial\Omega} \operatorname{tr}(G(z)^{-1}G'(z))\, dz. \tag{3.2}$$

which can be stochastically estimated by

$$s \approx \tilde{s} := \frac{1}{2\pi \mathrm{i} L} \int_{\partial\Omega} \sum_{k=1}^{L} u_j^* G(z)^{-1} G'(z) u_j,\, dz, \tag{3.3}$$

where u_j are i.i.d random vectors with entries in $\{0, 1\}$. Given that $\tilde{s}$ is indeed an estimation, we cannot be sure that $\tilde{s} \geqslant s$. Hence, every author that worked with contour integrals added a "safety valve" before extracting the eigenvalues of M [Bey12; Asa+10; BEG20]: if $B_0^{[m]}$ is not full rank, then the deficiency is caused by the matrix J and thus we will be able to retrieve the eigenpairs.

In the holomorphic case, if the estimation of $\tilde{s}$ is good enough, then it is quite rare for $B_0^{[m]}$ not to be rank-deficient. In the worst scenario, then one has to increase m and/or p and recompute the SVD of the new $B_0^{[m]}$. Nevertheless, this is one of the situations where $F(z)$ being a meromorphic function causes several difficulties. First of all, now (3.2) does not compute the number of eigenvalues, but the difference between the number of eigenvalues and the poles of the determinant. Hence, (3.2) may vastly underestimate the optimal m if these two quantities are approximately the same. In addition, as we had seen in Remark 2.1, the poles of $\det F(z)$ are generally a *strictly* subset of the poles of $F(z)$. Furthermore, the rank of $B_0^{[m]}$ is influenced not only by the original eigenvalues of $F(z)$, but also from its spurious eigenvalues.

To conclude, we can always build a meromorphic function $F(z)$ such that (3.2) does not give useful information about $\operatorname{rank}(B_0^{[m]})$. The only way to make sure we have chosen a suitable m is computing the SVD of $B_0^{[m]}$ and increase the number of moments m or the size of the probing matrix P if $B_0^{[m]}$ is not rank-deficient. Note that recomputing the decomposition from scratch is not necessary in either cases. Indeed, let $P \in \mathbb{C}^{n\times p}$ and let $B_0^{[m]}(P) \in \mathbb{C}^{mn\times mp}$ be the Hankel matrix $B_0^{[m]}$ where we underline the presence of P. In order to have a rank-deficient Hankel matrix, we can either increase the number of moments or we can increase the size of the probing matrix P. Let us say we choose the first option, i.e., we take $m' > m$ and compute A_k for $k = 2m-1, 2m, \dots, 2m'-2$. Then we can write

$$B_0^{[m']}(P) = \left[\begin{array}{ccc|ccc} & & & A_m & \dots & A_{m'-1} \\ & B_0^{[m]}(P) & & \vdots & & \vdots \\ & & & A_{2m-1} & \dots & A_{m+m'-2} \\ \hline A_m & \dots & A_{2m-1} & A_{2m} & \dots & A_{m+m'-1} \\ \vdots & & \vdots & \vdots & & \vdots \\ A_{m'-1} & \dots & A_{m+m'-2} & A_{m+m'-1} & \dots & A_{2m'-2} \end{array}\right] \in \mathbb{C}^{m'n\times m'p}$$

and $B_0^{[m']}(P)$ is obtained from $B_0^{[m]}(P)$ by adding $(m'-m)n$ rows and $(m'-m)p$ columns. Hence, we obtain $B_0^{[m']}$ from $B_0^{[m]}$ with a $(m'-m)(n+p)$-rank update. In general, we will add further moments when $p = n$, hence the rank update will be $2(m'-m)n$. When this quantity is not too large with respect to mn, then there exist efficient ways to update the SVD of $B_0^{[m]}$ to obtain the one of $B_0^{[m']}$ (see, for instance, [Bra06; HL08]). More precisely, Brand proved that one can perform a rank r update of a $u\times v$ SVD in $\mathcal{O}(uvr)$, if $r^2 \leqslant \min\{u, v\}$ [Bra06].

If we decide to increase the size of the probing matrix, the strategy is similar. Denote with $A_k(P)$ the k-th moment computed with P. Consider now a larger probing matrix $P' = [P\ \overline{P}] \in \mathbb{C}^{n \times p'}$, where $\overline{P} \in \mathbb{C}^{n \times (p'-p)}$. Then

$$B_0^{[m]}(P) = \begin{bmatrix} A_0(P) & \dots & A_{m-1}(P) \\ \vdots & \iddots & \vdots \\ A_{m-1}(P) & \dots & A_{2m-2}(P) \end{bmatrix}, \qquad B_0^{[m]}(P') = \begin{bmatrix} A_0(P') & \dots & A_{m-1}(P') \\ \vdots & \iddots & \vdots \\ A_{m-1}(P') & \dots & A_{2m-2}(P') \end{bmatrix}.$$

It is not too difficult to see that there exists a block-permutation matrix Π such that

$$\begin{aligned} \Pi B_0^{[m]}(P') &= \left[\begin{array}{ccc|ccc} A_0(P) & \dots & A_{m-1}(P) & A_0(\overline{P}) & \dots & A_{m-1}(\overline{P}) \\ \vdots & \iddots & \vdots & \vdots & \iddots & \vdots \\ A_{m-1}(P) & \dots & A_{2m-2}(P) & A_{m-1}(\overline{P}) & \dots & A_{2m-2}(\overline{P}) \end{array}\right] \\ &= \begin{bmatrix} B_0^{[m]}(P) & B_0^{[m]}(\overline{P}) \end{bmatrix}, \end{aligned}$$

Therefore adding $p' - p$ columns to P and then rebuilding $B_0(P')$ is equivalent to adding $(p' - p)m$ columns to $B_0(P)$. Once again, if $(p' - p)m$ is relatively small compared to pm, then we can directly update the SVD of $B_0(P)$, saving computational time. In Algorithm 3.1 we summarise the procedure to choose p and m. Each time we either increase p or m by one, but a generalisation is trivial. Furthermore, if the sizes of the matrices are small, then recomputing the SVD might be faster than updating.

Remark 3.1. In general, it is better to increase the size of the probing matrix P instead of the number of moments m when $p < n$. As we have seen, changing p so that mp increases requires a lower rank update on B_0. Furthermore, we will see in Section 3.4 that it is better for the backward error of the eigenpairs to have fewer moments m.

3.3.1 The computational cost

Every author who studied contour algorithms to solve eigenvalue problems (and we are not an exception) has always underlined their natural parallelizability. If we di-

Algorithm 3.1: Choose the parameters p and m so that $B_0^{[m]}(P)$ is rank-deficient its SVD.

```
   Input: $F \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n}), \Omega \subset \Omega_0, L$
   Output: Rank-deficient SVD of $B_0^{[m]}(P)$.
 1 Draw $L$ i.i.d vectors $u_i \in \{0,1\}^n$
 2 $\tilde{s} \leftarrow \frac{1}{2\pi i L}\int_{\partial\Omega} \mathrm{tr}(F(z)^{-1}F'(z))\,dz$
 3 $m \leftarrow 1$, $p \leftarrow 1$
 4 while $mp \leqslant \tilde{s}$ do
 5     $p \leftarrow \min\{n,\ \lceil(\tilde{s}+1)/m\rceil\}$
 6     if $mp \leqslant \tilde{s}$ then // $p = n$ and $m$ is not large enough
 7         $m \leftarrow m+1$
 8 Build random matrix $P \in \mathbb{C}^{n\times p}$
 9 Compute $A_k(P)$ for $k = 0,\dots,2m-1$ and build $B_0^{[m]}(P)$
10 Compute reduced SVD $B_0 = V_0\Sigma_0W_0^*$
11 while $\Sigma_0$ is full-rank do
12     if $p < n$ then
13         $p' \leftarrow p+1$
14     else
15         $m' \leftarrow m+1$
16         $p' \leftarrow \min\{n,\ m(p+1)/m'\}$
17     if $p' \leqslant p$ then
18         $P' \leftarrow P(:,1:p')$
19     else
20         Build random $P_1 \in \mathbb{C}^{n\times(p'-p)}$
21         $P' \leftarrow [P\ P_1]$
22     Update SVD of $B_0^{[m]}(P)$ to obtain SVD of $B_0^{[m']}(P')$
23     $P \leftarrow P'$, $p \leftarrow p'$, $m \leftarrow m'$
24 return $V_0$, $\Sigma_0$, $W_0$
```

vide Ω in k subregions $\Omega_1,\dots,\Omega_k$, then each eigenvalue problem is independent from the others; furthermore, the linear systems in the approximation of the moments A_k can be solved independently as well. In our current world, where parallel computing has become as important as (if not more than) sequential computing, these are fundamental qualities for an algorithm, and often overshadow traditional measures, such as the computational cost expressed in *flops*. Nevertheless, a rigorous analysis of this aspect helps us understand the sections of the algorithm that require more attention.

We analyse Beyn's original formulation (see Algorithm 2.2), but the similarities with its Loewner interpretation would allow us to draw similar conclusions time-wise. First of all, suppose that each integral is approximated with N quadrature points. The first step is the estimation of the number of eigenvalues minus poles. Following the notation of (3.3), this requires the solution of N $n \times n$ linear systems, where the right hand side is a $n \times L$ matrix. Depending on the specific scenario, many algorithms can be used to solve them. We assume that n is not too large and we can store the N LU factorisations, which implies that this step costs $N(2/3n^3 + L\mathcal{O}(n^2))$. Similarly, the approximation of each moment A_k costs $Np\mathcal{O}(n^2)$ and therefore building $B_0^{[m]}(P)$, $B_1^{[m]}(P)$ requires $2mNp\mathcal{O}(n^2)$. Notice how the cost in n is quadratic because we are using the fact that we have saved the LU factorisations. Finally, the SVD of $B_0^{[m]}(P) \in \mathbb{C}^{mn \times mp}$ costs $14mnm^2p^2 + 8m^3p^3 = 2m^3p^3(7n/p + 4)$ [Hig08]. Adding everything together yields

$$\frac{2N}{3}n^3 + 2m^3p^3\left(\frac{7n}{p} + 4\right) + N(1 + 2m + L + p)\mathcal{O}(n^2) + \text{cost of } \texttt{eig}. \tag{3.4}$$

At first sight, this estimation does not seem very helpful because, as seen in the previous paragraphs, p and m are chosen at runtime. Nevertheless, under the assumptions that all the eigenvalues are simple and that $c_i \geqslant m_{\xi_i,1}$ for any spurious eigenvalue ξ_i, we have $mp \gtrsim s$. Hence, if we substitute $mp = s$ in (3.4) and leave out the lower order terms we get

$$\frac{2N}{3}n^3 + 2s^3\left(\frac{7n}{p} + 4\right). \tag{3.5}$$

Equation (3.5) clarifies an important aspect of contour algorithms: in a general meromorphic eigenvalue problem, the two terms in (3.5) are independent from each other, as written with greater details in Section 1.2.1. This is not true for the simpler polynomial eigenvalue problem of degree d, where s is always bounded by nd. Admittedly, in most practical applications we will have $n \gg s$, but we cannot dismiss the scenarios where the opposite is true. Hence, the algorithm could be very computationally expensive even if n is small, if there are several eigenvalues in Ω. Finally, the same analysis works for the Loewner interpretation as well, providing we substitute the product mp with the number of sample points r: we would then substitute again r with s and arrive at the same formula, minus a constant factor.

Remark 3.2. In the analysis above we are also assuming that the cost of finding the "correct" m and p is negligible with respect to the other operations, which is equivalent to assuming that either m and p are known a priori or the estimation of the number of eigenvalues (3.3) is precise. An easy example where this is no longer true is when the number of poles almost coincides with the number of eigenvalues. Another possibility is when there are several spurious eigenvalues. Consider, for instance, the following function $F(z) \in \mathcal{M}(\mathbb{C}, \mathbb{C}^{n\times n})$

$$F(z) = \begin{bmatrix} z & h(z)^{-1} & & \\ & 1 & \ddots & \\ & & \ddots & h(z)^{-1} \\ & & & 1 \end{bmatrix}, \quad h(z) = \prod_{i=1}^{r} (z - \xi_i)$$

with $\xi_i \neq 0$ and $\xi_i \neq \xi_j$ for $i \neq j$. Assume it is given in black box form and that we are interested in finding the eigenvalue $\lambda = 0$. The estimation of the number of eigenvalues minus poles returns 1 (the poles ξ_i are not seen by $\det(F(z))$), but

$$\operatorname{rank}(B_0^{[m]}) = 1 + r(n-1),$$

due to its spurious poles. In the end, we will have to solve a linear eigenvalue problem of size $1 + r(n-1)$, even though $F(z)$ has only one eigenvalue.

3.4 THE INFLUENCE OF THE QUADRATURE RULES

Approximating the contour integral is surely the most important step in the homonym algorithms. The choice of the quadrature often falls into the trapezoidal rule. The reason is twofold: not only it is very easy to implement, but it also offers an exponential convergence to the true value of a closed integral when the integrand function is holomorphic in an open neighbourhood of $\partial\Omega$. See, for example, [TW14, Theorem 2.2] or [DR07]. Hence very few papers focus more on this point or more precisely on the choice of the number N of quadrature points. Nevertheless, the next example shows that it is a topic that requires the most careful scrutiny.

3.4.1 A motivating example

We consider the `butterfly` problem (see also Example 2.3) and the `hadeler` problem

$$\begin{aligned} G_1(z) &= z^4B_4 + z^3B_3 + z^2B_2 + zB_1 + B_0, \qquad && B_k \in \mathbb{R}^{64\times 64}, \\ G_2(z) &= (e^z - 1)H_2 + z^2H_1 - 100I, && H_k \in \mathbb{R}^{30\times 30} \end{aligned}$$

from the NLEVP library [Bet+11; Had67]. The matrices H_k, B_2 and B_4 are symmetric, while B_1 and B_3 are skew-symmetric. The respective target sets are

$$\Omega_1 = \mathcal{D}(1+3i, 4) = \mathcal{D}(\gamma_1, r_1), \qquad \Omega_2 = \mathcal{D}(0,5) = \mathcal{D}(\gamma_2, r_2).$$

The reason for the unusual choice of the target sets is that we want to avoid any possible symmetry in the two problems. We are looking for a relationship between the approximation of the moments A_k and the backward error of the eigenpairs. We measure these two quantities as follows. We define

$$A_k^{(N)} := \sum_{j=1}^{N} w_{j,i} \left(\frac{z_{j,i} - \gamma_i}{r_i}\right)^k F(z_{j,i})^{-1}P \approx \frac{1}{2\pi i}\int_{\partial\Omega_i} \left(\frac{z-\gamma_i}{r_i}\right)^k F(z)^{-1}P\,dz$$

to be the approximation of A_k with N trapezoidal quadrature points, where $(z_{j,i}, w_{j,i})_{j=1}^N$ are the quadrature points and weights for the contour $\partial\Omega_i$. Then, we set

$$E_{(N,2N)}(A_k) := \frac{\left\|A_k^{(2N)} - A_k^{(N)}\right\|_2}{\left\|A_k^{(2N)}\right\|_2}$$

to be the relative error between two approximations of the integrals. In addition, let

$$\eta(\lambda, v) = \frac{\|G(\lambda)v\|_2}{\|G\|_{\Omega_i}\|v\|_2},$$

be the backward error of an eigenpair (see Section 2.2.3), where we approximate $\|G\|_{\Omega_i}$ with

$$\|G\|_{\Omega_i} \approx \max_{1 \leqslant j \leqslant N} \|G(z_{j,i})\|_2.$$

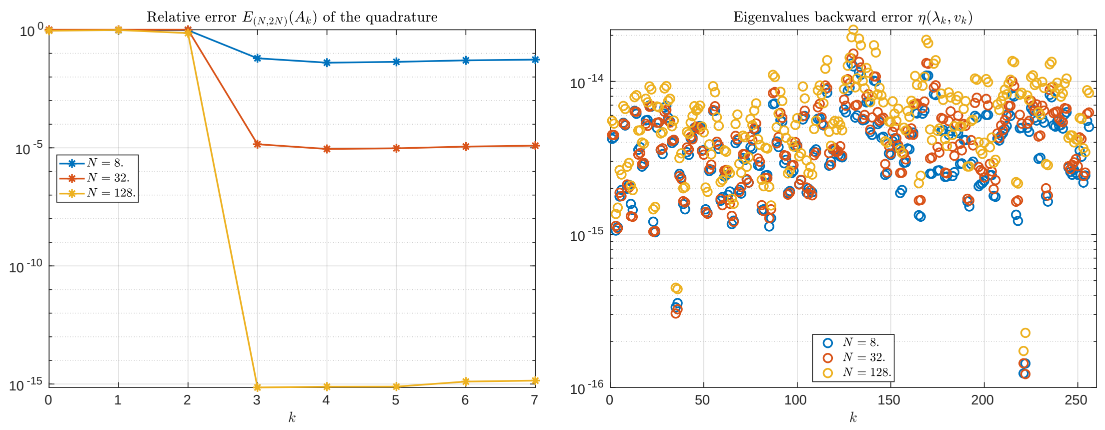
The set Ω_1 contains all the 256 eigenvalues of $G_1(z)$, hence we need to set $p = 64$ and $m = 4$, thus we compute A_k for $k = 0, \dots, 7$. In Figure 3.1a we plotted $E_{N,2N}(A_k)$ for $k = 0, \dots, 7$ and $N = 8, 32, 128$, while in Figure 3.1b the backward error $\eta(\lambda, v)$ for all the 256 eigenpairs and the respective values of N. We can see that even though the approximation of the moments is really rough for $N = 8$ and $N = 32$, the backward errors are already at machine precision. We point out that A_0, A_1, and A_2 are equal to 0, and that is why the relative error has order of unity. In fact, let $D_\rho = \mathcal{D}(0, \rho)$, then

$$\left\| \frac{1}{2\pi \mathrm{i}} \int_{\partial D_\rho} z^k G_1(z)^{-1} \right\| \lesssim \rho^{k+1-4} \to 0 \qquad \text{as } \rho \to \infty$$

if $k \leqslant 2$. The residue theorem thus tells us that the sum of all the residues, i.e., the eigenvalues of $G_1(z)$, is zero. The assertion follows from the fact that Ω_1 contains all the eigenvalues.

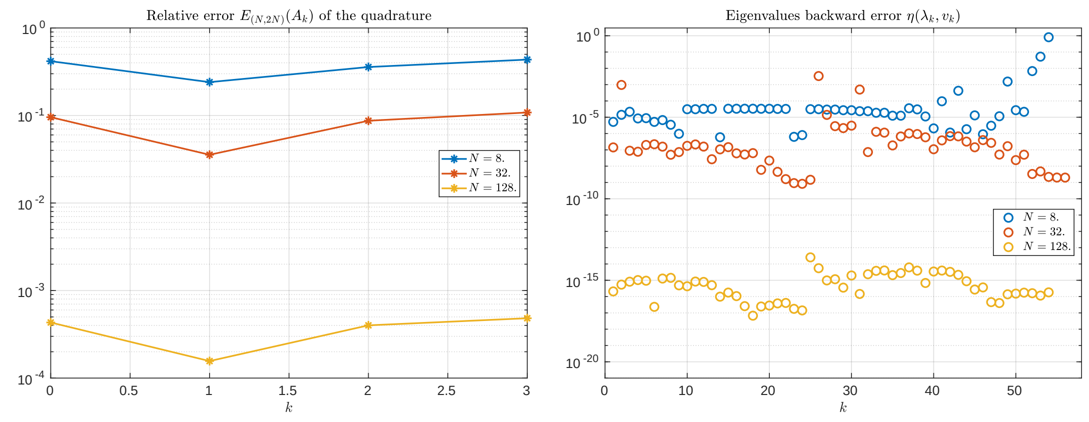
Remark 3.3. The observation above easily extends to rational functions. If $G(z)$ is a rational function of type (d_1, d_2) and Ω contains all the eigenvalues, then $A_k = 0$ for $k = 0, \dots, d_1 - d_2 - 2$.

The `hadeler` problem shows the other side of the coin. The set Ω_2 contains 54 eigenvalues, therefore we need to set $p = 30$ and $m = 2$. Figure 3.2 mirrors Figure 3.1: in 3.2a we plotted the quadrature error $E_{N,2N}(A_k)$, while in 3.2b the backward error of the eigenpairs. The differences are evident and staggering. The results for $N = 8$ and $N = 16$ are not accurate at all: in the first case, some backward errors are of the order of unity; in the latter, the algorithm even returns an incorrect number of eigenvalues. Among the three choices of N, only $N = 128$ returns a satisfactory output.



(a) **(b)**

Figure 3.1: Error of the approximation of the moments $A_k \in \mathbb{R}^{64\times64}$ (left) and backward error of the eigenpairs in Ω_1 (right) for the butterfly problem.



(a) **(b)**

Figure 3.2: Error of the approximation of the moments $A_k \in \mathbb{R}^{30\times30}$ (left) and backward error of the eigenpairs in Ω_1 (right) for the hadeler problem.

These two examples show that one should not underestimate the importance of the quadrature in the contour algorithms, even when they have the helpful property of being exponentially convergent. Indeed, in the first case a really rough approximation of the moments allows us to retrieve all the eigenpairs to machine precision. In the second case, the same rough approximation is not able to do the same. Therefore, we cannot solely rely on $E_{N,2N}(A_k)$ to set a "correct" number of quadrature points N, because we could either overestimate or underestimate this value. Nevertheless, a Gaussian-Kronrod quadrature rule is implemented in our algorithm, together with the more "standard" trapezoidal one.

3.4.2 The exponential error decay with the trapezoidal rule

There are noteworthy exceptions among authors who contributed to give us a better understanding of the influence of the quadrature approximations. Among them, Beyn gave a precise estimation of the exponential decay of the approximation error of A_p [Bey12]. We aim to generalise those results to the meromorphic settings.

Consider $\phi(t)\colon [0, 2\pi] \to \partial\Omega$ a 2π-periodic parametrization of the contour $\partial\Omega$. In addition, we assume it can be holomorphically extended on a strip

$$S(d_-, d_+) := \{z \in \mathbb{C}\colon -d_- < \mathrm{Im} z < d_+\}$$

such that

$$\phi(z) \begin{cases} \in \Omega, & 0 < \mathrm{Im} z < d_+, \\ \notin \Omega & -d_- < \mathrm{Im} z < 0. \end{cases} \tag{3.6}$$

Then Beyn proved the following results.

Lemma 3.1 ([Bey12, Lemma 4.6]). *Let $\phi(z)$ be a parametrization satisfying (3.6). In addition, let $\lambda \in \mathbb{C}$ and $h(z) := z - \lambda$. Then there exist three positive constants C_1, C_2, and C_3 (depending only on Ω, ϕ, j, and d) such that for $\operatorname{dist}(\lambda, \partial\Omega) < C_3$*

$$E_N\left(h(z)^{-j}\right) \leqslant C_1 \operatorname{dist}(\lambda, \partial\Omega)^{-j} \mathrm{e}^{-NC_2 \operatorname{dist}(\lambda,\partial\Omega)}.$$

Theorem 3.2 ([Bey12, Theorem 4.7]). *Let $G(z) \in \mathcal{H}(\Omega_0, \mathbb{C}^{n\times n})$ satisfying the Hypotheses 2.1. Assume there exists a parametrization $\phi(z)$ of $\partial\Omega$ satisfying (3.6). Finally, let $j = \max_{1\leqslant i\leqslant s} m_{i,1}$ be the largest index among the eigenvalues of $G(z)$. Then there exist positive constants C_1, C_2 (independent of N) such that the Hankel moments A_k and their approximations satisfy*

$$\left\| A_k - A_k^{(N)} \right\| \leqslant C_1 d(G)^{-j} \mathrm{e}^{-C_2 N d(G)},$$

where $d(G) = \min_{\lambda\in\Lambda(G)} \operatorname{dist}(\lambda, \partial\Omega)$

It is not too difficult to see that the previous results rely on the approximation of the scalar functions $h(z)^{-j} = (z-\lambda)^{-j}$ for j less or equal than the maximum index of the eigenvalues of $G(z)$ in Ω. This can be easily generalised to the meromorphic case $F(z) \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$ satisfying the usual Hypotheses 2.2. Remember that we can write

$$\begin{aligned} F(z)^{-1} &= \sum_{i=1}^{\tilde{s}} \sum_{j=1}^{d_i} \sum_{k=1}^{m_{ij}} (z-\lambda_i)^{-k} \sum_{\ell=0}^{m_{ij}} v_\ell^{ij} w_{m_{ij}-k-\ell}^{ij*} \\ &+ \sum_{i=1}^{\tilde{r}} \sum_{j=1}^{d_{\xi_i}} \sum_{k=1}^{m_{\xi,ij}} g(z)(z-\xi_i)^{-k} \sum_{\ell=0}^{m_{\xi,ij}} v_{\xi,\ell}^{ij} w_{\xi,m_{ij}-k-\ell}^{ij*}, \end{aligned}$$

hence the generalisation of Lemma 3.1 must consider the terms

$$\frac{g(z)}{(z-\xi_i)^k} = \prod_{j=1, j\neq i}^{r} (z-\xi_j)^{c_j} \frac{1}{(z-\xi_i)^{k-c_i}}.$$

More precisely, in the generalisation of Lemma 3.1 we would have both $(z-\lambda_i)^{-j}$ for $1 \leqslant j \leqslant m_{i,d_i}$, and $(z-\xi_i)^{c_i-j}$ for $1 \leqslant j \leqslant m_{\xi_i,d_{\xi_i}}$. The proof of the lemma does not change, therefore we refer to [Bey12] for the details. We can now generalise the statement of Theorem 3.2 so that it covers the case of $F(z) \in \mathcal{M}(\Omega, \mathbb{C}^{n\times n})$.

Theorem 3.3. *Let $F(z) \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$ with the spectral properties of Notation 2.2. Assume there exists a parametrization $\phi(z)$ of $\partial\Omega$ satisfying (3.6). Finally, let j be*

$$j = \max\{\widehat{j},\, \widetilde{j}\}, \qquad \widehat{j} = \max_{1\leqslant i\leqslant s} m_{i,1}, \quad \widetilde{j} = \max_{1\leqslant i\leqslant r}(m_{\xi_i,1} - c_i).$$

Then there exist positive constants C_1, C_2 (independent of N) such that the Hankel moments A_k and their approximations satisfy

$$\left\| A_k - A_k^{(N)} \right\| \leqslant C_1 d(F)^{-j} \mathrm{e}^{-C_2 N d(F)},$$

where $d(F) = \min\{\widehat{d}(F),\, \widetilde{d}(F)\}$, and

$$\widehat{d}(F) = \min_{1\leqslant i\leqslant s} \operatorname{dist}(\lambda_i, \partial\Omega), \qquad \widetilde{d}(F) = \min_{1\leqslant i\leqslant r, m_{\xi_i,1} > c_i} \operatorname{dist}(\xi_i, \partial\Omega).$$

Example 3.1. *In this example we consider the functions*

$$F_1(z) = \begin{bmatrix} z & 0 & 0 & 0 \\ 0 & (z-\xi)^{-1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad F_2(z) = \begin{bmatrix} z & (z-\xi)^{-1} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$F_3(z) = \begin{bmatrix} z & (z-\xi)^{-1} & 0 & 0 \\ 0 & 1 & (z-\xi)^{-1} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad F_4(z) = \begin{bmatrix} z & (z-\xi)^{-1} & 0 & 0 \\ 0 & 1 & (z-\xi)^{-1} & 0 \\ 0 & 0 & 1 & (z-\xi)^{-1} \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

in the target set $\Omega = \mathcal{D}(0,1)$ and we investigate how the distance of the pole ξ from $\partial\Omega$ influences the forward error of the only eigenvalue ($\lambda = 0$) in Ω. It is easy to see that the pole multiplicity of ξ is always $c = 1$, while its geometric multiplicity as an eigenvalue of $G_i(z) = (z-\xi)F_i(z)$ is $d_\xi = 1$ and its index $m_{\xi,1} = i$. We sampled 100 uniformly spaced points ξ in $[0.5, 0.997]$ and we computed the eigenvalue $\lambda = 0$ through the contour integral

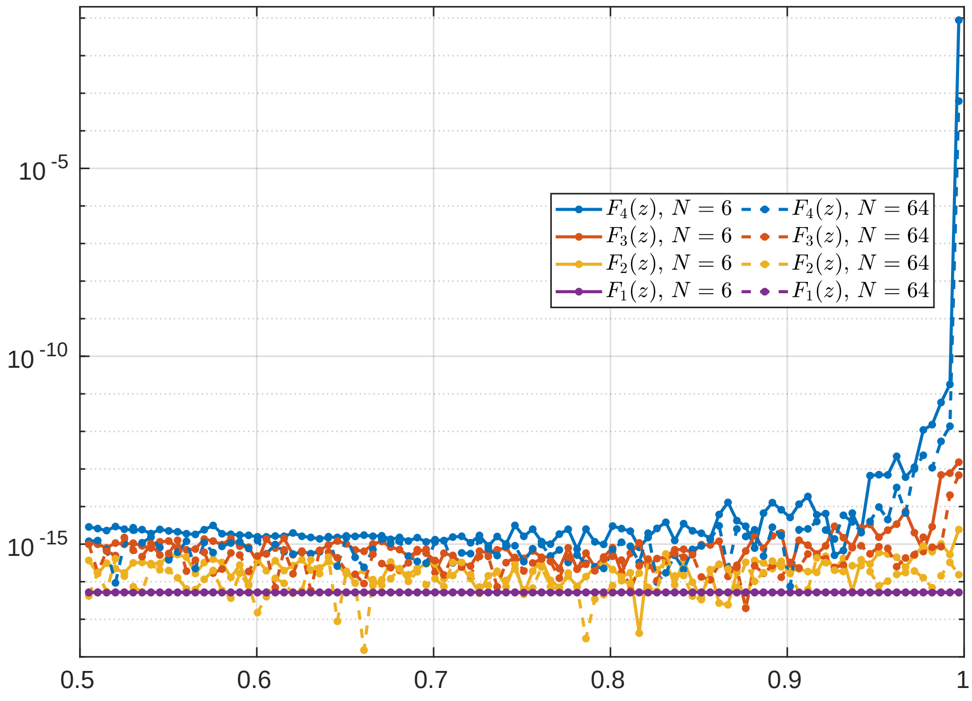


Figure 3.3: The forward error of the approximation $\widehat{\lambda}$ to the eigenvalue $\lambda = 0$ for different values of the pole $\xi \in [0.5, 1[$.

algorithm for all the values of ξ. In Figure 3.3 we plotted these results for $N = 6$ and $N = 64$ quadrature points. As explained by Theorem 3.3, we have $d(F_i) = (1 - \xi)^{1-i}$, hence the larger the index of the spurious eigenvalue ξ, the larger the error will be. Finally, for $i = 1$ we have $d(F_1) = 1$ and in fact the forward error of $\lambda = 0$ does not depend on ξ.

3.4.3 Quadrature approximations as filter functions

In 2016 Van Barel and Kravanja proposed a different point of view on the quadrature approximations [Van16; VK16]. They link the approximation of $A_p^{(N)}$ to the construction of a so-called *filter function* $b_p(z)$ whose absolute value decays exponentially outside Ω. More recently, Brennan, Embree, and Gugercin found a similar filter function $b_\sigma(z)$ in the Loewner case, but they did not provide an analysis for it yet, as far as we know [BEG20]. The goal of this section is understanding if anything changes when we consider a meromorphic function $F(z) \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$.

First, let us recall the first steps walked by Van Barel and Kravanja in the holomorphic case. In this section we need stronger hypotheses on Ω_0. They assume Ω_0 is simply connected, open, bounded and that there exists a holomorphic function $T \in \mathcal{H}(\Omega_0, \mathbb{C}^{n\times n})$ (more precisely, in an open neighbourhood of Ω_0); furthermore, they assume $T(z)$ has s simple eigenvalues in Ω and no eigenvalues neither on $\partial\Omega$ nor $\partial\Omega_0$. Thanks to a quadrature rule with N points $z_0, \dots, z_{N-1}$ and weights $\omega_0, \dots, \omega_{N-1}$, they rewrite

$$A_p^{(N)} = \sum_{j=0}^{N-1} \omega_j z_j^p T(z_j)^{-1}. \tag{3.7}$$

The core of the analysis is applying Keldysh's theorem *not on* Ω, but on Ω_0. Assuming there are $\tilde{s}$ simple eigenvalues in Ω_0, (3.7) becomes

$$\sum_{j=0}^{N-1} \omega_j z_j^p \sum_{k=1}^{\tilde{s}} \frac{v_k w_k^*}{z_j - \lambda_k}. \tag{3.8}$$

Equations (3.7) and (3.8) already reveal the difficulties that arise when considering a meromorphic function $F \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$. In fact, when defining the holomorphization $G(z) := g(z)F(z) \in \mathcal{H}(\Omega, \mathbb{C}^{n\times n})$ we cannot assume that it is holomorphic in Ω_0 as well, i.e., $F(z)$ has only poles in Ω. Given that our intention is applying Keldysh theorem on both Ω and Ω_0, we have to work with three functions:

- $F(z)$, which may have poles in Ω and Ω_0;
- $G(z) := g(z)F(z)$, the holomorphization of $F(z)$ in Ω, which may have poles only in $\Omega_0 \backslash \Omega$;
- $T(z) := t(z)G(z) = t(z)g(z)F(z)$, the holomorphization of $F(z)$ in Ω_0.

Now that we have clarified the need of these three functions, we can lay down the other assumptions for this section. As in the article by Van Barel and Kravanja, we suppose that all the eigenvalues, both the original of $F(z)$ and the spurious ones, are simple. We also assume that $F(z)$ has s ($\widetilde{s}$) eigenvalues and r ($\widetilde{r}$) poles in Ω (Ω_0), and that all the poles are spurious eigenvalues. Under these hypotheses, it holds

$$A_p = VJ^pW^* = \sum_{k=1}^{s} \frac{z^{p-1} v_k w_k^*}{z-\lambda_k},$$

where v_k, w_k are the eigenvectors of $F(\lambda_k)$ subjected to the condition

$$w_k^* F(\lambda_k)' v_k = 1. \tag{3.9}$$

Note that the spurious eigenvalues ξ_k in J disappear because $g(\xi_k) = 0$. Thus we have

$$A_p \approx A_p^{(N)} = \sum_{j=0}^{N-1} \omega_j z_j^p F(z_j)^{-1}. \tag{3.10}$$

Now we apply Keldysh theorem for meromorphic functions (see Theorem 2.7) on $F(z)$ in Ω_0. Hence

$$F(z)^{-1} = \sum_{k=1}^{\widetilde{s}} \frac{g(z)t(z)}{z-\lambda_k} \widetilde{v}_k \widetilde{w}_k^* + \sum_{k=1}^{\widetilde{r}} \frac{g(z)t(z)}{z-\xi_k} \widetilde{v}_{\xi_k} \widetilde{w}_{\xi_k}^* + R(z), \tag{3.11}$$

where $\widetilde{v}_k, \widetilde{w}_k$ and $\widetilde{v}_{\xi_k}, \widetilde{w}_{\xi_k}$ are the eigenvectors corresponding to λ_k and ξ_k respectively, subject to the conditions

$$\begin{cases} \widetilde{w}_k^* T(\lambda_k)' \widetilde{v}_k & = 1 \Longleftrightarrow \widetilde{w}_k^* F(\lambda_k)' \widetilde{v}_k = g(\lambda_k)^{-1} t(\lambda_k)^{-1}, \\ \widetilde{w}_{\xi_k}^* T(\xi_k)' \widetilde{v}_{\xi_k} & = 1. \end{cases} \tag{3.12}$$

By substituting (3.11) into (3.10) we get

$$\begin{aligned} A_p^{(N)} = & \sum_{k=1}^{\widetilde{s}} \widetilde{v}_k \widetilde{w}_k^* \sum_{j=0}^{N-1} \frac{\omega_j z_j^p g(z_j) t(z_j)}{z_j - \lambda_k} \\ & + \sum_{k=1}^{\widetilde{r}} \widetilde{v}_{\xi_k} \widetilde{w}_{\xi_k}^* \sum_{j=0}^{N-1} \frac{\omega_j z_j^p g(z_j) t(z_j)}{z_j - \xi_k} + \sum_{j=0}^{N-1} \omega_j R(z_j). \end{aligned} \tag{3.13}$$

Before continuing, we need a technical lemma.

Lemma 3.4 ([VK16, Section 3]). *Let $g(z) \in \mathbb{C}[z]$ be a polynomial and let $z_j = e^{2\pi i j/N}$ be the N unit roots for $j = 0, \dots, N-1$. Then*

$$\frac{1}{N}\sum_{j=0}^{N-1} \frac{z_j g(z_j)}{z_j - z} = \frac{g(z)}{1 - z^N}.$$

Proof. First of all, note that we have to prove the result only for a general monomial $g(z) = z^p$. In fact, if that is the case, then for any polynomial $g(z) = \sum_{k=0}^{p} g_k z^k$ we have

$$\begin{aligned}\frac{1}{N}\sum_{j=0}^{N-1} \frac{z_j g(z_j)}{z_j - z} &= \frac{1}{N}\sum_{j=0}^{N-1} \frac{\sum_{k=0}^{p} g_k z_j^{k+1}}{z_j - z} = \sum_{k=0}^{p} \frac{1}{N}\sum_{j=0}^{N-1} \frac{g_k z_j^{k+1}}{z_j - z} \\ &= \sum_{k=0}^{p} \frac{g_k z^k}{1 - z^N} = \frac{g(z)}{1 - z^N}.\end{aligned}$$

Hence, we set $g(z) = z^p$. It is easy to see that

$$\frac{1}{N}\sum_{j=0}^{N-1} \frac{z_j^{p+1}}{z - z_j} = \frac{p(z)}{z^N - 1},$$

for some polynomial $p(z)$ yet to be determined. It follows that

$$Np(z) = \sum_{j=0}^{N-1} z_j^{p+1} \prod_{\substack{k=0 \\ k \neq j}}^{N-1} (z - z_k).$$

We have to prove that $p(z) = z^p$. Fix $0 \leqslant j' \leqslant N-1$, Then

$$Np(z_{j'}) = z_{j'}^{p+1} \prod_{\substack{k=0 \\ k \neq j}}^{N-1} (z_{j'} - z_k) = z_{j'}^{p+N} \prod_{\substack{k=0 \\ k \neq j'}}^{N-1} (1 - z_{k-j'}) = z_{j'}^{p+N} \prod_{k=1}^{N-1} (1 - z_k).$$

Since $z_k = e^{2\pi i k/N}$, then

$$\prod_{k=1}^{N-1} (1 - z_k) = \left.\frac{z^N - 1}{z - 1}\right|_{z=1} = N.$$

Hence $p(z_j) = z_j^p$ for $0 \leqslant j \leqslant N-1$, and given that $p(z)$ is at most a $N-1$ degree polynomial, $p(z) = z^p$. □

If we choose the trapezoidal rule and $\Omega_0 = \mathcal{D}(0,1)$, then the quadrature points and the weights are $z_j = \mathrm{e}^{2\pi \mathrm{i} j/N}$ and $\omega_j = z_j$, respectively. Therefore, thanks to Lemma 3.4, Equation(3.13) becomes

$$\begin{aligned} A_p^N &= \sum_{k=1}^{\widetilde{s}} \widetilde{v}_k \widetilde{w}_k^* \frac{1}{N} \sum_{j=0}^{N-1} \frac{z_j^{p+1} g(z_j) t(z_j)}{z_j - \lambda_k} + \sum_{k=1}^{\widetilde{r}} \widetilde{v}_{\xi_k} \widetilde{w}_{\xi_k}^* \frac{1}{N} \sum_{j=0}^{N-1} \frac{z_j^{p+1} g(z_j) t(z_j)}{z_j - \xi_k} + \frac{1}{N} \sum_{j=0}^{N-1} z_j R(z_j) \\ &= \sum_{k=1}^{\widetilde{s}} \widetilde{v}_k \widetilde{w}_k^* \frac{\lambda_k^p g(\lambda_k) t(\lambda_k)}{1 - \lambda_k^N} + \sum_{k=1}^{\widetilde{r}} \widetilde{v}_{\xi_k} \widetilde{w}_{\xi_k}^* \frac{\xi_k^p g(\xi_k) t(\xi_k)}{1 - \xi_k^N} + \frac{1}{N} \sum_{j=0}^{N-1} z_j R(z_j) \\ &= \sum_{k=1}^{\widetilde{s}} \widetilde{v}_k \widetilde{w}_k^* \frac{\lambda_k^p g(\lambda_k) t(\lambda_k)}{1 - \lambda_k^N} + \frac{1}{N} \sum_{j=0}^{N-1} z_j R(z_j), \end{aligned} \tag{3.14}$$

where in the third equality the second sum disappears because $g(\xi_k) t(\xi_k) = 0$ for $k = 1, \dots \widetilde{r}$.

Remark 3.4. The assumptions on the eigenvalues and spurious eigenvalues of $F(z)$ to be semisimple are essential to arrive to (3.14). In the general case, we would have

$$F(z)^{-1} = g(z) \sum_{i=1}^{\widetilde{s}} \sum_{j=1}^{d_i} \sum_{k=1}^{m_{ij}} (z - \lambda_i)^{-k} \sum_{\ell=0}^{m_{ij}} v_\ell^{ij} w_{m_{ij}-k-\ell}^{ij*} + \sum_{i=1}^{\widetilde{r}} \sum_{j=1}^{d_{\xi_i}} \sum_{k=1}^{m_{\xi,ij}} (z - \xi_i)^{-k} \sum_{\ell=0}^{m_{\xi,ij}} v_{\xi,\ell}^{ij} w_{\xi,m_{ij}-k-\ell}^{ij*},$$

and we could not use Lemma 3.4 or any reasonable generalisation.

Due to the normalisation conditions (3.9) and (3.12), it holds

$$\begin{aligned} A_p^N &= \sum_{k=1}^{\widetilde{s}} \widetilde{v}_k \widetilde{w}_k^* \frac{\lambda_k^p g(\lambda_k) t(\lambda_k)}{1 - \lambda_k^N} + \frac{1}{N} \sum_{j=0}^{N-1} z_j R(z_j) \\ &= \sum_{k=1}^{\widetilde{s}} v_k w_k^* \frac{\lambda_k^p}{1 - \lambda_k^N} + \frac{1}{N} \sum_{j=0}^{N-1} z_j R(z_j) \\ &= \sum_{k=1}^{\widetilde{s}} v_k w_k^* b_p(\lambda_k) + \frac{1}{N} \sum_{j=0}^{N-1} z_j R(z_j), \end{aligned} \tag{3.15}$$

where we set

$$b_p(z) = \frac{z^p}{1 - z^N} \tag{3.16}$$

to be the *Hankel rational filter function* for $\Omega = \mathcal{D}(0,1)$, which is the exact function defined by Van Barel and Kravanja [VK16]. As expected, under the hypotheses that the spurious eigenvalues disappear, we obtain the same result of the holomorphic case. Their analysis then continues and they define

$$\Omega_\varepsilon(b_0) := \{z \in \mathbb{C} : |b_0(z)| \geqslant \varepsilon\} \tag{3.17}$$

for $\varepsilon \ll 1$. If s_ε is the number of eigenvalues in $\Omega_\varepsilon(b_0)$, and we reorder the eigenvalues such that

$$|b_0(\lambda_1)| \geqslant \cdots |b_0(\lambda_{s_\varepsilon})| \geqslant |b_0(\lambda_{s_\varepsilon+1})| \geqslant \cdots \geqslant |b_0(\lambda_{\tilde{s}})|$$

then one can rewrite

$$A_p^N = \sum_{k=1}^{s_\varepsilon} v_k w_k^* b_p(\lambda_k) + \Delta_1 + \Delta_2, \qquad \text{with } \|\Delta_i\|_2 = \mathcal{O}(\varepsilon^{1-p/n}), \quad i = 1,2. \tag{3.18}$$

This analysis gives an explanation of the examples of section 3.4.1: we saw that we were able to retrieve the eigenvalues of the `butterfly` problem with very few quadrature points because there were no eigenvalues outside the target set; on the other hand, the computed moments for the `hadeler` problem suffered more from the noise of the eigenvalues outside Ω.

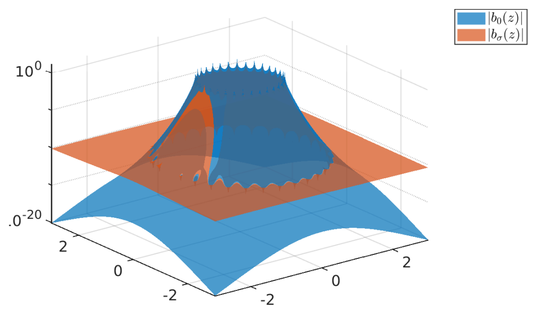
Brennan, Embree, and Gugercin performed the first steps of this analysis for the Loewner framework [BEG20]. They showed that (2.43) is approximated by

$$H^N(\sigma) = \sum_{k=1}^{\tilde{s}} w_k b_\sigma(\lambda_k) + \frac{1}{N}\sum_{j=0}^{N-1} \frac{R(z_j)}{\sigma - z_j},$$

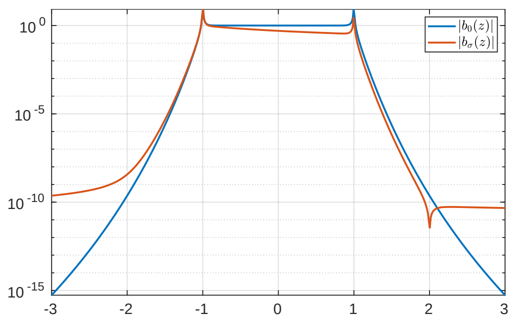
where

$$b_\sigma(z) = \frac{1}{\sigma - z}\left(\frac{1}{1-z^N} - \frac{1}{1-\sigma^N}\right),$$

with $\sigma \notin \Omega = \mathcal{D}(0,1)$. The same analysis Van Barel and Kravanja performed for Beyn's algorithm [VK16] can be performed with $b_\sigma(z)$ and one obtains the equivalent



(a) $|b_0(z)|$ and $|b_\sigma(z)|$ on $\mathbb{C}$.



(b) $|b_0(z)|$ and $|b_\sigma(z)|$ on $\mathbb{R}$.

Figure 3.4: Absolute values of the filter functions $b_0(z)$ and $b_\sigma(z)$ with $N = 32$, $\sigma = 2$.

of (3.18) for $H^N(\sigma)$. However, things get more interesting when we investigate the shape of $\Omega_\varepsilon(b_0)$ and $\Omega_\varepsilon(b_\sigma)$ in (3.17). It is easy to see that $\Omega_\varepsilon(b_0)$ is approximated by the disk $D = \mathcal{D}(0, \varepsilon^{-1/N})$ when $\varepsilon \ll 1$. On the other hand, the shape of $\Omega_\varepsilon(b_\sigma)$ requires a more careful analysis. Under the assumption that $|\sigma| > 1$, $|z| > 1$, the first order approximation of $b_\sigma(z)$ yields

$$\frac{|\sigma^N - z^N|}{|\sigma - z||\sigma z|^N} = \varepsilon.$$

This functions has four different main regimens

- If $\sigma \approx z$, then $|b_\sigma(z)| = \mathcal{O}(Nz^{-N-1})$.
- If $|\sigma^N - z^N| = \delta \ll 1$, then $|b_\sigma(z)| = \mathcal{O}(|\sigma - z|^{-1}\sigma^{-2N}\delta)$.
- If $|z| < |\sigma|$,then $|b_\sigma(z)| = \mathcal{O}(|\sigma - z|z^{-N})$.
- If $|z| > |\sigma|$,then $|b_\sigma(z)| = \mathcal{O}(|\sigma - z|\sigma^{-N})$.

In Figure 3.4 we have plotted the graphs of the functions $|b_0(z)|$ and $|b_\sigma(z)|$ to highlight their differences. We set $N = 32$ and $\sigma = 2$. We can see their behaviour on the complex plane in 3.4a, and on the real line in plot 3.4b. It is easy to see that while $b_\sigma(z)$ decreases slightly more steeply for small values of $|z| > 1$, as soon as $|z| > |\sigma|$, the decay becomes linear. For example, if we set $\varepsilon = 10^{-13}$, then $\Omega_\varepsilon \approx \mathcal{D}(0, 2.548)$, but $\Omega_\varepsilon \approx \mathcal{D}(0, \rho)$, with $\rho = \varepsilon/\sigma^N \approx 2380$. Hence many more eigenvalues contribute to the noise of H^N_σ than to the noise of A^N_p.

The next example shows that the errors on A_p^N and H_σ^N is reflected in the backward error of the eigenvalues. Therefore, if one desires to compute the eigenvalues of a function $F(z)$ with the Loewner algorithm and a backward error less than ε, then they should choose the interpolation points σ_i such that

$$|\sigma_i| < \varepsilon^{-1/N}.$$

Noting that for a general disk $\Omega = \mathcal{D}(\gamma, \rho)$ we have

$$b_\sigma(z) = \frac{1}{\sigma - z}\left(\frac{1}{1-\left(\frac{z-\gamma}{\rho}\right)^N} - \frac{1}{1-\left(\frac{\sigma-\gamma}{\rho}\right)^N}\right),$$

then the condition becomes

$$|\sigma - \gamma| < \rho\varepsilon^{-1/N}. \tag{3.19}$$

Example 3.2. *In this example we show the influence of the outer eigenvalues with respect to the final backward error of the computed eigenpairs. We consider the very simple matrix-valued functions $G_j(z) = A_j - zI$ for $j = 1, \ldots, 5$. The matrices $A_j \in \mathbb{C}^{40\times 40}$ are built such that they all have 20 eigenvalues uniformly distributed in $[-0.9, 0.9]$, while the other 20 eigenvalues lie on a circle of radius r_j, where $r_j \in \{1.5, 2.5, 4, 6, 100\}$. We used $N = 32$ trapezoidal points on the unit circle to retrieve the eigenvalues in the unit disk. The sampling points σ_i for the Loewner method lay on the circle of radius 2.*

In Figure 3.5 we show the influence of the outer eigenvalues on the backward error. In the plots on the left we plotted the backward errors of the eigenpairs of G_j for the Hankel (top-left) and the Loewner case (bottom-left); on the right we reported the graphs of $|b_0(z)|$ and $|b_\sigma(z)|$ from Figure 3.4b. We witness the following behaviours:

1. *When the outer eigenvalues lie on $r_1 = 1.5$, Hankel and Loewner return comparable backward errors of magnitude approximately equal to 10^{-5}.*

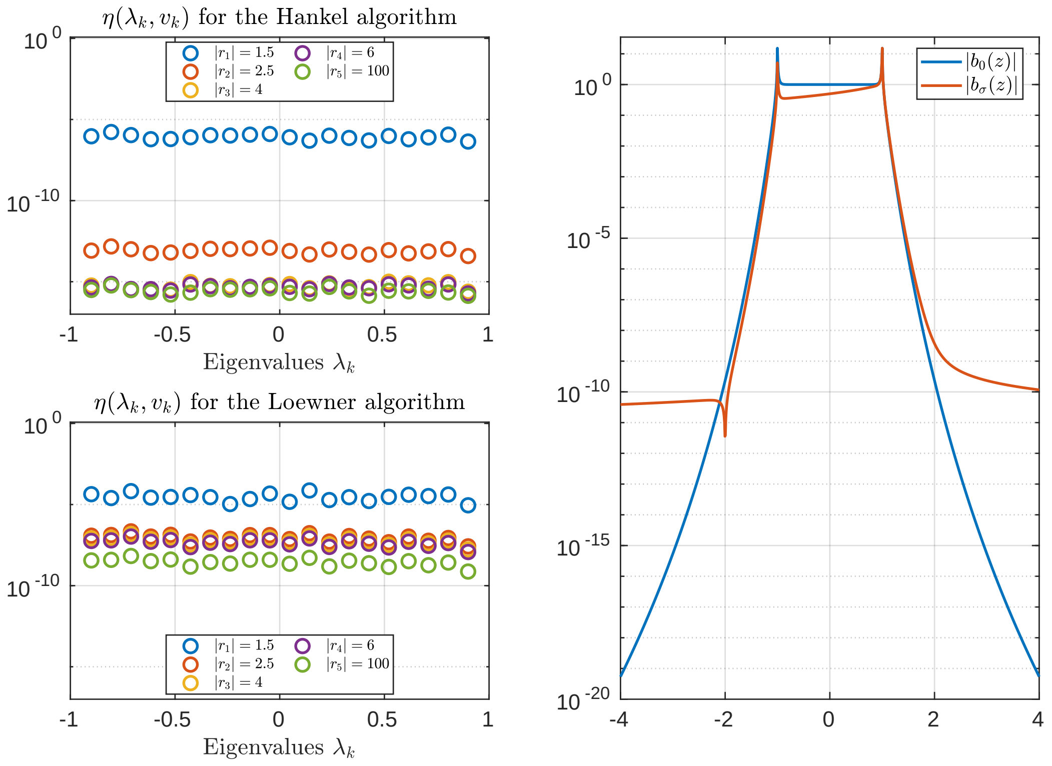


Figure 3.5: Left plot: Backward errors of the eigenpairs computed by the Hankel algorithm (top) and the Loewner algorithm (bottom) for different positions of the outer eigenvalues. Right plot: the profile of $|b_0(z)|$ and $|b_\sigma(z)|$.

2. *In the Hankel case, the backward errors keep dropping exponentially as the outer eigenvalues move away from the target set, until they reach machine precision for* $r_3 = 4$ *and* $r_4 = 6$.

3. *In the Loewner case, we have the exponential drop for* $r_2 = 2.5$*, but then the decay is only linear, given that the interpolation points* σ_i *satisfy* $|\sigma_i| = 2$*. In fact, we have to set* $r_5 = 100$ *to notice the decay.*

As expected, the decay of the backward errors follows the one of the filter functions. Finally, it is important to point out that the linear decay in the Loewner case can easily be avoided: one just needs to choose the interpolation points σ_i *and the number of quadrature points* N *such that* (3.19) *holds true for* ε *equal to the machine precision. In this example, drawing the* σ_i *from the circle of radius* 3 *would have sufficed.*

Remark 3.5. As far as we know, even though the previous analysis follows naturally from the work of Van Barel and Kravanja [VK16] applied on the Loewner framework [BEG20], the fact that one needs to choose the interpolation points σ_i and the number of quadrature points N such that (3.19) holds true is still missing in the current literature.

3.5 THE REFINEMENT STRATEGIES

In the introduction of this chapter, we stated that our main goal is proposing an algorithm to efficiently compute the eigenpairs of small and medium-sized meromorphic matrix-valued functions $F(z) \in \mathcal{M}(\Omega_0, \mathbb{C}^{n\times n})$, ideally asking the user to specify only $F(z)$ and Ω. It may happen that the automatic choice of the parameters at runtime leads to some eigenpairs having a backward error larger than the given threshold ε. In order to avoid to restart the algorithm from scratch we propose three refinement strategies, each of them suited for a different scenario.

Let us fix some notation specifically for this section. We assume that all the eigenvalues are simple and that there exist s_b eigenpairs $(\widetilde{\lambda}_i, \widetilde{v}_i)$ which have not been computed accurately enough, i.e.,

$$\eta(\widetilde{\lambda}_i, \widetilde{v}_i) = \frac{\left\| F(\widetilde{\lambda})\widetilde{v}_i \right\|_2}{\|\widetilde{v}_i\|_2} > \varepsilon \tag{3.20}$$

for $i = 1, \dots, s_b$ and a given $\varepsilon > 0$. We refer to Section 2.2.3 for the reason why we have to choose an absolute measure for the backward error. For the sake of convenience, we call these eigenpairs "bad eigenpairs". The refinement strategy to be adopted heavily depends on the number s_b: if s_b is small, then the strategy would differ from when s_b is large.

3.5.1 Recursive calls

The first refinement we analyse is a recursive strategy already used in the literature, such as in the FEAST algorithm [GMP18] and naturally by RIM algorithm (see Section 2.3). If we assume that the s_b eigenvalues are clustered in K subregions, we can create K non overlapping subregions Ω_i for $i = 1, \dots, K$ and solve K independent subproblems. Note that the independence of the problems means that this kind of refinement can be easily parallelised. A few questions now arise:

- Where should the algorithm put the subset Ω_i?
- How do we set K if it is unknown?
- When is this strategy convenient?

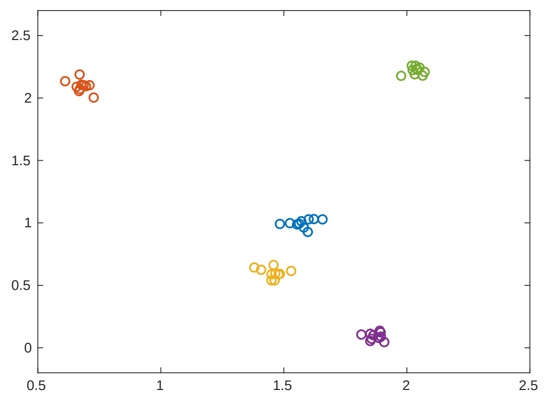
The first question is the easiest to answer, but it requires the definition of the *K-means clustering problem*.

Definition 3.1 (K-means clustering problem)**.** Given a set of points $\Lambda = \{\lambda_1, \dots, \lambda_{s_b}\} \subset \mathbb{C}$ and an integer number $1 < K < s_b$, the K-means clustering problem requires the partition of Λ in K subsets $(S_i)_{i=1}^K$ of n_i elements such that

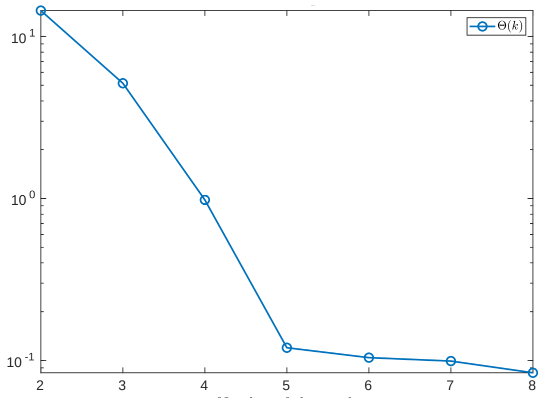
$$\Theta(k) = \sum_{i=1}^{K} \sum_{j=1}^{n_i} \left|\lambda_{i,j} - \mu_i\right|^2, \qquad \mu_i = \sum_{j=1}^{n_i} \frac{\lambda_{i,j}}{n}$$

is minimised, where μ_i are the centroids of the clusters.

When K is unknown, the K-means clustering problem is NP-hard, even in the planar case [GJW82; MNV09]. Nevertheless, when K is given, there exist heuristic algorithms that converge to a local minimum in $\mathcal{O}(Ks_b)$ almost always [AV06]. This answers how to place the subsets Ω_i, but not how to choose K. If the user is supervising the algorithm at runtime, then they can choose K by looking at the plot of the unrefined eigenvalues. Nevertheless, we wanted another way to find K without outside assistance and we decided to use the so-called "elbow-method". First of all, observe that Θ_K is decreasing with respect to K and it drastically drops as soon as K is equal to the exact number of clusters of the problem, i.e., the "elbow". Hence, we can run the K-means algorithm for different values of K and then choose the optimal one. For example, in Figure 3.6a we plotted 50 points in the plane subdivided in 5 clusters; in Figure 3.6b we plotted the quantity $\Theta(k)$ with respect to k: the typical



(a) Five clusters of points in the plane.



(b) The typical "elbow" drop of the error $\Theta(k)$.

Figure 3.6

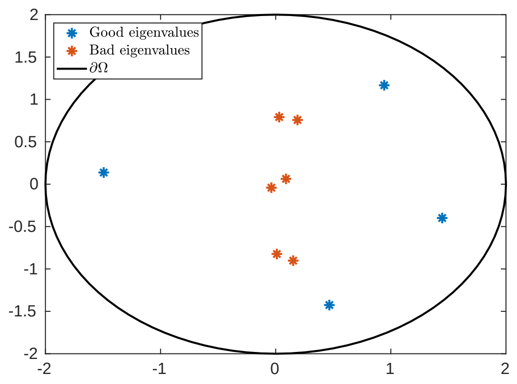
"elbow" shape is clearly visible, therefore it is reasonable to assume simply from 3.6b itself that there are 5 clusters.

In the next example we show how the recursive refinement can improve the output of the algorithm under some specific circumstances.

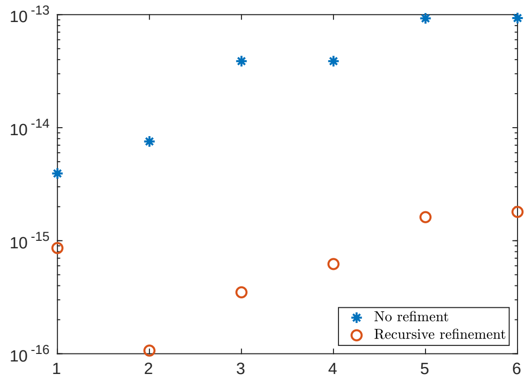
Example 3.3. *We built a nonlinear problem $F(z)$ with three clusters of eigenvalues in the disks $D(\gamma_i, 10^{-2})$ for $\gamma_i \in \{-0.8\mathrm{i}, 0, 0.8\mathrm{i}\}$. Each cluster contains two randomly drawn eigenvalues, and four other randomly drawn eigenvalues lie on the circle of radius 1.5, for a total of ten eigenvalues in $\Omega = \mathcal{D}(0,2)$. Furthermore, we added a pole of double multiplicity at each γ_i. The problem $F(z)$ is equivalent to*

$$D(z) = \mathrm{diag}(p_1(z), p_2(z), p_3(z), g(z), q(z)^{-2})$$

where $p_i(z)$ is a quadratic polynomial whose roots are the eigenvalues clustered in Ω_i, $g(z)$ is a quartic polynomial whose roots are the other eigenvalues, while $q(z)$ is a cubic polynomial with roots γ_i. In Figure 3.7a we plotted the 10 eigenvalues: we highlighted in red the ones that belong to the three clusters and lie near the poles of $F(z)$, which have a worse forward error than the eigenvalues in blue. In Figure 3.7b we plotted the forward error $|\lambda_i - \widetilde{\lambda}_i|/|\lambda_i|$, where λ_i are the exact clustered eigenvalues, while $\widetilde{\lambda}_i$ the approximation returned by our algorithm, with and without the recursive refinement. The algorithm run with 16 quadrature points for the initial contour, 6 moments, and a threshold on the backward error equal to $\varepsilon = 10^{-15}$.



(a) The eigenvalues of $F(z)$ in Ω.



(b) The forward error of the bad eigenvalues.

Figure 3.7

The recursive refinement has some downsides as well. For instance, if a cluster of bad eigenvalues contains eigenvalues that were approximated well, we have to make sure to keep the best approximation among the original one and the one computed during the recursion. Similarly, if the clusters are not too spaced, it may happen that the $\Omega_i \cap \Omega_j \neq \emptyset$, since we are working with circles and ellipses, hence we may refine the same eigenvalue more than once. Finally, the refinement itself could be more expensive than the original run: assume a "good" eigenvalue λ_i lies just outside a given Ω_i. Then we will need to approximate the contour integrals with many quadrature points N', and this quantity is unrelated to the number of quadrature points N used on the original set Ω. For all the reasons mentioned above, the current implementation of our solver adopts this refinement if and only if the user requires it.

3.5.2 The Newton refinement

In the introduction of the thesis we recalled that the algorithms to solve nonlinear eigenvalue problems are usually based on rational approximations (Chapter 4), contour integrals (this chapter), and Newton-like methods. We also wrote that this latter class has two main disadvantages: first, we need a starting point near each eigenpair, because the convergence is only local; secondly, it is quite costly if there are many eigenpairs to compute. Therefore it is natural to adopt this strategy if only a few eigenpairs have a large backward error. In addition, since $(\widetilde{\lambda}_i, \widetilde{v}_i)$ is already an approximation of the desired eigenpair, a couple of steps are usually sufficient to reach the desired backward error.

Let us now be more precise. Consider $(\widetilde{\lambda}_i, \widetilde{v}_i)$ the approximation of (λ_i, v_i) and assume that $\mathcal{N}\colon \mathbb{C}^n \times \mathbb{C} \to \mathbb{C}^n \times \mathbb{C}$ is a smooth function such that $\mathcal{N}(v_i, \lambda_i) = 0$. Then each Newton step reads

$$\begin{bmatrix} v_i^{k+1} \\ \lambda_i^{k+1} \end{bmatrix} = \begin{bmatrix} v_i^{k} \\ \lambda_i^{k} \end{bmatrix} - \left(J_{\mathcal{N}} \left(\begin{bmatrix} v_i^{k} \\ \lambda_i^{k} \end{bmatrix} \right) \right)^{-1} \mathcal{N} \left(\begin{bmatrix} v_i^{k} \\ \lambda_i^{k} \end{bmatrix} \right), \tag{3.21}$$

where $(\lambda_i^0, v_i^0) = (\widetilde{\lambda}_i, \widetilde{v}_i)$ and $J_{\mathcal{N}}$ is the Jacobian of $\mathcal{N}$. Ruhe analysed this approach in [Ruh73], where he set

$$\mathcal{N}\left(\begin{bmatrix} v \\ \lambda \end{bmatrix} \right) = \begin{bmatrix} F(\lambda)v \\ u^* v - 1 \end{bmatrix}, \quad J_{\mathcal{N}} \begin{bmatrix} v \\ \lambda \end{bmatrix} = \begin{bmatrix} F(\lambda) & F'(\lambda)v \\ u^* & 0 \end{bmatrix}$$

with $u \in \mathbb{C}^n$ is a given vector. We can expand (3.21) in a scalar form as

$$\begin{cases} v_i^{k+1} = F(\lambda_i^k)^{-1}(\lambda_i^{k+1} - \lambda_i^k) F'(\lambda_i^k) v_i^k, \\ u^* v_i^{k+1} = 1. \end{cases} \tag{3.22}$$

The scalar form (3.22) is equivalent to the nonlinear inverse iteration [Ung50], which we summarise in Algorithm 3.2. Note that when λ^k is near λ, then $F(\lambda^k)$ is nearly singular. Nevertheless, it is now well-known that the error in the computed vector v_i^{k+1} is almost parallel to the real solution (see, for example, [Ips97]), hence a normalisation of v_i^{k+1} at every step will correct this error and also prevent underflow and overflow.

In order to avoid computing the same eigenpair (λ_i, v_i) more than once, we need a deflation strategy. In our numerical experiments, the initial guess $(\widetilde{\lambda}_i, \widetilde{v}_i)$ provided by the core of the contour solver is always good enough so that we return the corresponding eigenpair (and not one previously computed), however a solid deflation strategy adds another layer of robustness. We decided to adopt the one proposed

Algorithm 3.2: Nonlinear Inverse iteration. The algorithm takes as input the function $F(z)$, an approximated eigenpair $(\widetilde{\lambda}, \widetilde{v})$, a threshold ε, a nonzero vector u, and the maximum number of iterations M. It returns the refined eigenpair (λ, v).

```
   Input: $F, \widetilde{\lambda}, \widetilde{v}, u, \varepsilon, M$
   Output: Refined eigenpair $(\lambda, v)$.
1  $(\lambda_0, v_0) \leftarrow (\widetilde{\lambda}, \widetilde{v})$ $k \leftarrow 0$, $\delta \leftarrow \varepsilon + 1$
2  while $k < M$ and $\delta > \varepsilon$ do
3      Find $v^{(k+1)}$ such that $F(\lambda^{(k)})v^{(k+1)} = F'(\lambda^{(k)})v^{(k)}$
4      $\lambda^{(k+1)} \leftarrow \lambda^{(k)} - \frac{u^* v^{(k)}}{u^* v^{(k+1)}}$
5      $v^{(k+1)} \leftarrow v^{(k+1)} / \|v^{(k+1}\|$
6      $k \leftarrow k + 1$
7      $\delta \leftarrow \|F(\lambda^{(k+1)})v^{(k+1)}\|$
8  return $(\lambda^{(k)}, v^{(k)})$
```

by Effenberger in [Eff13], which we briefly describe. In Definition 2.4 we introduced the concept of minimal pair and minimality index to understand the minimum size of the matrices $B_0^{[m]}$, $B_1^{[m]}$. Effenberger assumes we have computed a m-minimal pair $(V, J) \in \mathbb{C}^{n \times i} \times \mathbb{C}^{i \times i}$ and he wants to extend it to

$$(\overline{V}, \overline{J}) := \left(\begin{bmatrix} V & w \end{bmatrix}, \begin{bmatrix} J & v \\ 0 & \mu \end{bmatrix} \right)$$

In order to do that we can solve a larger nonlinear eigenvalue problem

$$0 = \overline{F}(\mu)\overline{v} = \begin{bmatrix} F(\mu) & U(\mu) \\ A(\mu) & B(\mu) \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix}, \tag{3.23}$$

where

$$A(\mu) = \sum_{k=0}^{m} \mu^k (VJ^k)^*, \quad U(\mu) = \frac{1}{2\pi \mathrm{i}} \int_{\partial\Omega} F(\mu) V (zI - J)^{-1} (z - \mu)^{-1} \, dz,$$
$$B(\mu) = \sum_{k=1}^{m} (VJ^k)^* V q_k(\mu), \quad q_k(\mu) = \sum_{j=0}^{k-1} \mu^j J^{k-j+1}.$$

This result was proven in the following theorem.

Theorem 3.5 ([Eff13, Theorem 3.8])**.** *Let $(V, J) \in \mathbb{C}^{n\times i} \times \mathbb{C}^{i\times i}$ be a minimal pair for $F(z)$. If $([y^* \ v^*]^*, \mu)$ is a minimal pair of* (3.23)*, then*

$$\left(\begin{bmatrix} V & w \end{bmatrix}, \begin{bmatrix} J & v \\ 0 & \mu \end{bmatrix} \right)$$

is a minimal pair for $F(z)$. Conversely, if

$$\left(\begin{bmatrix} V & w \end{bmatrix}, \begin{bmatrix} J & v \\ 0 & \mu \end{bmatrix} \right)$$

is a minimal pair of $F(z)$, then there exists a unique vector u such that

$$\left(\begin{bmatrix} w - Vu \\ v + \mu u - Ju \end{bmatrix}, \mu \right)$$

is a minimal pair of (3.23)

This strategy allows to refine successive eigenpairs without worrying to recompute a previous one. The difference between Effenberger's and our implementation is that we call it only when necessary: as previously mentioned, our initial approximation is meant to be good enough such that the Newton iteration will converge directly to the desired eigenpair. We summarise it here below and in Algorithm 3.3. We assume that we have s_b eigenpairs $(\widetilde{\lambda}_i, \widetilde{v}_i)_{i=1}^{s_b}$ to refine and we perform the following cycle:

1. At the ith step, refine $(\widetilde{\lambda}_i, \widetilde{v}_i)$ with the classic Newton iteration (3.22) and obtain $(\widehat{\lambda}_i, \widehat{v}_i)$. If $(\widehat{\lambda}_i, \widehat{v}_i) \neq (\widehat{\lambda}_j, \widehat{v}_j)$ for every $j \leqslant i-1$, then set $(\lambda_i, v_i) = (\widehat{\lambda}_i, \widehat{v}_i)$, $(V_i, J_i) = (v_i, \lambda_i)$ and go to step 1.

2. If $(\widehat{\lambda}_i, \widehat{v}_i) = (\widehat{\lambda}_{j'}, \widehat{v}_{j'})$ for some $j' \leqslant i-1$, then use Theorem 3.5 to extract an augmented minimal pair that we save as the new $(V_{j'}, J_{j'})$ and a new eigenpair $(\widehat{\lambda}_i, \widehat{v}_i)$. If $(\widehat{\lambda}_i, \widehat{v}_i)$ is different from any other eigenpair, then go to step 1, otherwise go to step 2.

In the implementation, discerning whether two eigenpairs $(\widehat{\lambda}_1, \widehat{v}_1)$ and $(\widehat{\lambda}_2, \widehat{v}_2)$ differ requires careful attention. In order to do that, let $|[\widehat{v}_1]_k| = \|\widehat{v}_1\|_\infty$, where $[\widehat{v}_1]_j$ is the jth component of the vector $\widehat{v}_1$. Then, define

$$\widetilde{v}_1 := \frac{[\widehat{v}_2]_k}{|[\widehat{v}_2]_k|} \frac{|[\widehat{v}_1]_k|}{[\widehat{v}_1]_k} \widehat{v}_1,$$

and recall that $\|\widehat{v}_i\|_2 = 1$. We set a tolerance ε and we say that the two eigenpairs are the same if $\left|\widehat{\lambda}_1 - \widehat{\lambda}_2\right| < \varepsilon \left|\widehat{\lambda}_1\right|$ and $\|\widetilde{v}_1 - \widehat{v}_2\|_2 < \varepsilon$. Note that the definition of $\widetilde{v}_1$ is necessary to avoid the scenario where $\widehat{v}_1$ and $\widehat{v}_2$ differ because $\widehat{v}_1 = \alpha \widehat{v}_2$, with $|\alpha| = 1$.

Algorithm 3.3: NEWTON REFINEMENT. The algorithm takes as input the function $F(z)$, the approximated eigenpairs $(\widetilde{\lambda}_i, \widetilde{v}_i)$, a threshold ε, and a maximum number of iterations. It then returns the refined eigenpairs (λ_i, v_i).

```
   Input: $F$, $(\widetilde{\lambda}_i, \widetilde{v}_i)_{i=1}^{s_b}$, $\varepsilon$, $M$
   Output: Refined eigenpairs $(\lambda_i, v_i)_{i=1}^{s_b}$ .
1  for $1 \leqslant i \leqslant s$ do
2      Draw a random vector $u$
3      $(\widehat{\lambda}_i, \widehat{v}_i) =$ NONLINEAR INVERSE ITERATION$(F, \widetilde{\lambda}_i, \widetilde{v}_i, u, \varepsilon, M)$ (Alg. 3.2)
4      $j \leftarrow 1$
5      while $j < i$ do
6          if $(\widehat{\lambda}_i, \widehat{v}_i) = (\lambda_j, v_j)$ then
7              Use Thm. 3.5 to update $(V_j, J_j)$
8              Update $(\widehat{\lambda}_i, \widehat{v}_i)$ from $(V_j, J_j)$
9              $j \leftarrow 1$
10         else
11             $j \leftarrow j + 1$
12     $(\lambda_i, v_i) \leftarrow (\widehat{\lambda}_i, \widehat{v}_i)$
13     $(V_i, J_i) \leftarrow (v_i, \lambda_i)$
```

3.5.3 The subspace update

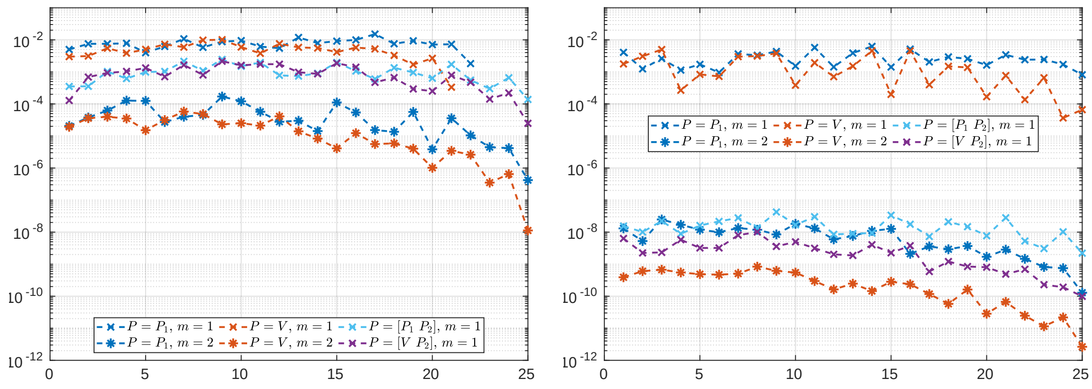
If s_b is too large, using the Newton method to refine each eigenpair becomes too expensive. Hence, we implemented a second type of refinement that takes inspiration from the work of Asakura and Sakurai [Asa+10]. In a sense, we introduced it in Section 3.3, when we described how to choose the suitable size of the matrix $B_0^{[m]}(P)$

if the function $F(z)$ is meromorphic. The approach here is identical. Given the projection matrix $P \in \mathbb{C}^{n \times p}$ and the number of moments m, if the eigenpairs obtained from $B_0^{[m]}(P)$ and $B_1^{[m]}(P)$ have a large backward error, then we can increase the size of the block-Hankel matrices and recompute the eigenpairs. If $p < n$, then we would just choose $p < p' \leqslant n$; if $p = n$, then one must increase the number of moments. In the following examples we show how we can apply this refinement.

Example 3.4. *We consider the* `pdde_symmetric` *problem*

$$F(z) = (M + A) - zI + \mathrm{e}^{-2z}B$$

from the NLEVP *library, where the matrices M, A, I are real of size 81×81. It arises from the discretization of a partial delay differential equation with Dirichlet boundary conditions. We look for the 25 eigenvalues in $\Omega = \mathcal{D}(0, 1.5)$ and we want to see how the refinement on P and m influences the backward error $\eta(\lambda, v)$ of the eigenpairs. We compare them with another refinement approach that seems reasonable at first glance, i.e., using the approximated eigenvectors $V \in \mathbb{C}^{81\times 25}$ as a new subspace matrix P. Let $P_1 \in \mathbb{C}^{81\times 25}$ and $P_2 \in \mathbb{C}^{81\times 12}$ be uniformly distributed random matrices. In Figure 3.8 we plotted the backward errors of the eigenpairs under six choices: the four combinations of $P \in \{V, P_1\}$ and $m \in \{1, 2\}$; the choice of $m = 1$ and $P = [V\ P_2]$ or $P = [P_1\ P_2]$. In 3.8a we have set $N = 32$ trapezoidal integration points, while in 3.8b $N = 128$. Note that the algorithm would not increase the number of moments until P becomes a square matrix, due to the computational costs, but we chose these settings to provide further insights. The first thing we note is that with $N = 32$ points and $m = 1$, the algorithm misses one eigenvalue if a random matrix is used (blue crosses) or even two when we refine with the eigenvector matrix V (red crosses). We point out that under the default parameters the algorithm would use a larger probing subspace, because $A_0(P_1)$ and $A_0(V)$ would be full-rank, hence it would not miss any eigenvalue. In addition, we can see that using two moments (the asterisk markers in the plots) produces the best results. This should not come as a surprise, because $B_0^{[2]}(P)$ has twice the rows and columns of $B_0^{[1]}(P) = A_0$. Furthermore, we see that increasing the size of the probing subspace (light*



(a) $N = 32$. **(b)** $N = 128$.

Figure 3.8: The backward error of the 25 eigenpairs of the `pdde_symmetric` problem with different kind of refinements.

blue crosses) provides a noticeable improvement of the backward errors for $N = 32$*, and a very good one for* $N = 128$*, comparable to add another moment (asterisk markers).*

Now that we have described the refinement strategies, we have to explain how the algorithm chooses which one to implement. As seen in Section 3.5.1, it opts for the recursive one only if the user specifically requests for it, given its specificity. We already mentioned that modifying the probing matrix P (or adding additional moments A_k) works best when s_b is large, while the Newton refinement shines if only few eigenvalues need to be refined. Therefore we want to compute an estimation of the numbers of floating point operations each refinement would do and then choose the lowest one. It is important to point out that our choice is based on our current MATLAB implementation, which does not take advantage of parallelisation.

We denote with $\mathcal{C}_n$ the computational cost of the Newton refinement, and with $\mathcal{C}_p$ the computational cost of updating the probing space. For the Newton method, we assume that the deflation strategy is not needed and that two or three steps are sufficient to reach the desired backward error. Hence we have

$$\mathcal{C}_n \approx 3n^3 s_b. \tag{3.24}$$

If we increase the size of the matrix $B_0^{[m]}(P)$, we can either add p' columns to the matrix P if $m = 1$, or add m' moments if $p = n$. In the first case, we spend $2Nn^2p'$ flops to compute the moments $A_j(P')$ for $j = 0, 1$, because we have saved the LU decomposition of $F(z_k)^{-1}$ for $k = 0, \dots, N-1$. Furthermore, if $p'^2 \leqslant p + p'$, then updating the SVD of A_0 costs $\mathcal{O}(np'(p+p'))$ [Bra06]. On the other hand, if we add m' moments we spend $2Nn^2pm'$ flops to compute them and $\mathcal{O}(n^3m'(m+m')^2)$ to update the SVD of $B_0^{[m]}$ when $(m'n)^2 \leqslant (m+m')n$. Finally, in both cases we spend $\mathcal{O}(s^3)$ flops to solve the linear eigenvalue problem. Summing everything up we obtain

$$\mathcal{C}_p \approx \begin{cases} 2Nn^2p' + np'(p+p') + s^3 & \text{if we add } p' \text{ columns to } P, \\ 2Nn^2pm' + n^3m'(m+m')^2 + s^3 & \text{if we add } m' \text{ moments.} \end{cases} \tag{3.25}$$

Therefore, if the user does not explicitly set one of the refinements proposed in this section, we compare $\mathcal{C}_n$ and $\mathcal{C}_p$ and choose the one with the lower value. We are aware that we have computed only an approximation of the true values of $\mathcal{C}_n$ and $\mathcal{C}_p$, but this would only matter when they are approximately equal, hence the choice between one or the other is irrelevant (time-wise) in that circumstance. Finally, we point out that the Newton refinement will always be the last stage of the algorithm, while the update of the probing subspace can be applied more than once. This implies first that we have to set a maximum number of times we can update the subspace, and secondly that it may happen that we first refine by updating the subspace and then by calling the Newton method.

3.6 FINAL REMARKS

The main goal of Chapter 2 was showing that they can be used to solve meromorphic eigenvalue problems, even though up to now authors have limited themselves to holomorphic ones. In many cases, an everyday user who invokes these algorithms will not notice whether $F(z)$ does or does not have poles in Ω. However, the theory we developed is necessary to understand the behaviour of the solvers in all the circumstances. In addition, we showed how minor changes can lead to big improvements to the Recursive Integral Method and we gave a probabilistic estimation of its computational cost in the nonlinear case. Future works on this matter have two clear directions. The first one is going from the abstract settings, which were our main topic, to more concrete applications. More specifically, one may consider difficult problems coming from physics or structural engineering. The `buckling_plate` problem in NLEVP 4.0 has indeed this origin: it was the simplification of a challenging application sent to us by Dr. Melina Freitag. The second road is continuing our path of abstraction. For instance, we did not consider the degenerate case in (2.7), where a point λ is both an eigenvalue and a pole for $F(z)$. Another possibility is investigating whether the results for meromorphic functions hold in the nonsquare case, as Morikuni did in 2020 for holomorphic functions [Mor20].

In Chapter 3 we focused more on the practical aspects of the implementation. First, we underlined the importance of having a collection of various problems to test our algorithms, which lead to the release of a newer version of the NLEVP library. Then, we analysed how the presence of poles in Ω influences the approximation of the contour integral, and we continued the research started by Van Barel and Kravanja [VK16] on the filter functions for the Loewner interpretation of Beyn's algorithm. This led to a relationship between the number of quadrature points N and the distance among the sampling points σ_j and the contour $\partial\Omega$. Finally, we implemented two refinement steps which the solver relies on when the automatic choice of the parameters does not lead to satisfactory results. We point out that this chapter does not contain a numerical experiments section because we will compare our contour solver at the end of Chapter 4 with the algorithms presented therein. We have two other directions in mind for further research. The first is either looking for even better, more sophisticate ways to choose the initial parameters, or developing an algorithm where the final approximation of the eigenvalues can be always refined without restarting from the beginning. The second lies more on the technical side. Our code, as most research code in the field, is written in MATLAB. If we ever want to have a fast and performing program, we should translate it in another language, such as C++ or Julia. As far as we are aware, the FEAST algorithm is the only famous C++ implementation of a contour solver, but evidently it does not incorporate yet the results we discovered in this thesis.

4 ROBUST RATIONAL APPROXIMATIONS OF NONLINEAR EIGENVALUE PROBLEMS

As we explained in Section 1.2.2, a possible method to solve the nonlinear eigenvalue problem for $G(z) \in \mathcal{H}(\Omega_0, \mathbb{C}^{n\times n})$ in $\Omega \subset \Omega_0$ is providing a rational approximation $R(z)$ such that $G(z) \approx R(z)$ in some sense and then solving the simpler rational eigenvalue problem, often by any linearization technique. We can write the rational approximant as

$$R^{(m)}(z) = b_0(z)R_0 + b_1(z)R_1 + \cdots + b_m(z)R_m \tag{4.1}$$

on the *target set* $\Omega \subset \Omega_0$. The $R_j \in \mathbb{C}^{n\times n}$ in (4.1) are constant-coefficient matrices and the b_j are polynomials of degree at most m or rational functions of type (m, m), that is, quotients of polynomials of degree at most m.

The main goal of this chapter is the numerical construction of a rational approximant $R(z)$ of $G(z)$ that is robust, i.e.,

- it is reliable for any given tolerance ε;
- it is scale-independent: if $R(z)$ is an approximant of $G(z)$, then $\alpha R(z)$ is an approximant of $\alpha G(z)$, for any $\alpha \in \mathbb{C}$.

More specifically, we use a discrete finite set $\Omega \supset \Sigma := \{\sigma_0, \dots, \sigma_M\}$, which is usually a fine mesh of Ω or simply some randomly drawn points and we will propose some algorithms that return a rational approximant $R^{(m)}(z)$ such that

$$\left\| G - R^{(m)} \right\|_\Sigma \leqslant \varepsilon \left\| R^{(m)} \right\|_\Sigma, \tag{4.2}$$

where we recall

$$\|G\|_\Sigma := \sup_{z\in\Sigma} \|G(z)\|_2 = \max_{z\in\Sigma} \|G(z)\|_2. \tag{4.3}$$

In this chapter we change the hypothesis of Ω being open (see Section 1.1) to close. This means that $\partial\Omega \subset \Omega$ and that Ω is compact. Further, we will focus on both the cases where $G(z)$ is provided in split form

$$G(z) = \sum_{j=1}^{s} g_j(z) A_j, \tag{4.4}$$

with $A_j \in \mathbb{C}^{n\times n}$ and $g_j(z) \in \mathcal{H}(\Omega_0, \mathbb{C})$, and the case where $G(z)$ is provided as a black box that only returns evaluations $G(z_0)$ for $z_0 \in \Omega$. The structure is as follows. In Section 4.1 we provide the error analysis for the eigenpairs of $G(z)$ computed through a rational approximant $R^{(m)}(z)$; in Section 4.2 we give a brief overview of modern algorithms that inspired us in developing our solution; in Section 4.3 we explain our two-phase algorithm that satisfies (4.2); Section 4.4 is dedicated to the numerical experiments that compare the proposed approximation algorithms with the state-of-the-art on the problems from NLEVP 4.0 (see Section 3.2). Finally, in Section 4.5 we compare the eigenvalues retrieved with the best algorithms of the previous section and the ones from the contour solver described in Chapters 2 and 3.

4.1 ERROR ANALYSIS OF APPROXIMATED EIGENPAIRS

Let $G(z)\colon \Omega_0 \to \mathbb{C}^{n\times n}$ and let $\Omega \supset \Sigma$ be the compact target set and its discrete mesh. In addition, let $R^{(m)}(z)$ be any rational approximant such that (4.2) is satisfied. In general, the relative error will not be bounded on Ω as well, but we expect something similar to hold if the functions $G(z)$ and $R^{(m)}(z)$ are "smooth" enough. More precisely, we expect that

$$\|G - R\|_{\Omega} \leqslant c_{\Omega} \varepsilon \|G\|_{\Omega} \tag{4.5}$$

holds true for some constant $c_{\Omega} > 1$. Indeed, if $G(z)$ and $R^{(m)}(z)$ are uniformly continuous on Ω, we have by [Che98, Lemma 2, p. 86] that

$$\left\| G - R^{(m)} \right\|_{\Omega} \leqslant \omega_G(\delta) + \omega_{R^{(m)}}(\delta) + \varepsilon \|G\|_{\Omega},$$

where

$$\omega_G(\delta) = \sup_{|z_1 - z_2| \leqslant \delta} \|G(z_1) - G(z_2)\|_2$$

is the modulus of continuity of $G(z)$ (and likewise for $R^{(m)}(z)$) and

$$\delta = \max_{z \in \Omega} \min_{\sigma \in \Sigma} |\sigma - z|$$

is the "density" of Σ in the target set Ω. By choosing δ such that

$$\omega_G(\delta) + \omega_{R^{(m)}}(\delta) \leqslant \varepsilon \|G\|_{\Sigma}$$

for a given ε, then (4.5) holds for $c_{\Omega} = 2$. This means that if the functions are uniformly continuous on Ω, then we can control the error on Ω with the error on the discretized set Σ, provided that δ is small enough, i.e., that Σ is dense enough in Ω.

A stronger argument holds when $G(z)$ and $R^{(m)}(z)$ are holomorphic in Ω. In this case, Σ can simply be a discretization of $\partial\Omega$ and (4.2) implies

$$\left\| G - R^{(m)} \right\|_{\partial\Omega} \leqslant \varepsilon c_{\partial\Omega} \|G\|_{\Omega},$$

where $c_{\partial\Omega} > 1$ is another positive constant. By the maximum norm principle, it follows $\|G\|_{\Omega} = \|G\|_{\partial\Omega}$ and therefore

$$\left\| G - R^{(m)} \right\|_{\Omega} \leqslant \varepsilon c_{\partial\Omega} \|G\|_{\Omega}.$$

In Section 2.2.3 we saw how to compute the backward error $\eta(\widehat{\lambda}, \widehat{v})$ of an eigenpair of $G(z)$ in (2.19). Now assume that (4.5) holds true. What is the backward error of the eigenpairs of $R^{(m)}(z)$ used as eigenpairs of $G(z)$? If $(\widehat{\lambda}, \widehat{v})$ is a computed eigenpair of a rational approximant $R^{(m)}(z)$ that satisfies (4.5) with backward error $\eta_{R^{(m)}}(\widehat{\lambda}, \widehat{v})$, then the backward error with respect to $G(z)$ is given by

$$\begin{aligned}
\eta_G(\widehat{\lambda}, \widehat{v}) = \frac{\left\| G(\widehat{\lambda})\widehat{v} \right\|_2}{\|G\|_\Omega \|\widehat{v}\|_2} &= \frac{\left\| G(\widehat{\lambda})\widehat{v} - R^{(m)}(\widehat{\lambda})\widehat{v} - \Delta R^{(m)}(\widehat{\lambda})\widehat{v} \right\|_2}{\|G\|_\Omega \|\widehat{v}\|_2} \\
&\leqslant \frac{\|G - R^{(m)}\|_\Omega}{\|G\|_\Omega} + \frac{\|\Delta R^{(m)}\|_\Omega}{\|G\|_\Omega} \\
&\leqslant c_\Sigma \varepsilon + \frac{\|R^{(m)}\|_\Omega}{\|G\|_\Omega} \, \eta_{R^{(m)}}(\widehat{\lambda}, \widehat{v}),
\end{aligned} \tag{4.6}$$

where in the first equality we used the definition of backward error (2.18).

In practice, if $(\widehat{\lambda}, \widehat{v})$ with $\widehat{\lambda} \in \Omega$ is a computed eigenpair of $R^{(m)}(z)$ with backward error $\eta_{R^{(m)}}(\widehat{\lambda}, \widehat{v}) \leqslant \varepsilon$ then we can expect $(\widehat{\lambda}, \widehat{v})$ to be an approximate eigenpair of $G(z)$ with a backward error $\eta_G(\widehat{\lambda}, \widehat{v}) \lesssim \varepsilon$ when $\|R^{(m)}\|_\Omega / \|G\|_\Omega \approx 1$ and c_Σ is not too large.

As already mentioned in Remark 2.3, the formula $\eta_G(\widehat{\lambda}, \widehat{v})$ is not practical, due to the presence of $\|G\|_\Omega$, hence we have to substitute it with a lower bound. In the numerical experiments of this chapter we will use

$$\widehat{\eta}_G(\widehat{\lambda}, \widehat{v}) := \frac{\left\| G(\widehat{\lambda})\widehat{v} \right\|_2}{\|G\|_\Sigma \|\widehat{v}\|_2}. \tag{4.7}$$

This affects (4.6) by a factor $\|G\|_\Omega / \|G\|_\Sigma \geqslant 1$.

Example 4.1. *We consider the* 2×2 *matrix-valued function* `nep1` *from the* NLEVP *library*

$$G(z) = \begin{bmatrix} e^{iz^2} & 1 \\ 1 & 1 \end{bmatrix}, \tag{4.8}$$

with eigenvalues

$$\lambda_{1,2} = 0, \quad \lambda_3 = \sqrt{2\pi}, \quad \lambda_4 = i\sqrt{2\pi}, \quad \lambda_5 = -i\sqrt{2\pi}, \quad \lambda_6 = -\sqrt{2\pi}$$

in the target set $\Omega = \mathcal{D}(0,3)$*. We generate* 1000 *random points in* Ω *and* 200 *points uniformly distributed on* $\partial\Omega$ *as the discrete set* Σ*. Then we build four rational approximants of* $G(z)$ *on* Σ *with different relative errors and degrees* m*—see the first two columns*

Table 4.1: Backward errors for approximate eigenpairs of $G(z)$ in (4.8) computed as eigenpairs of $R^{(m)}(z)$.

m	$\frac{\|G-R^{(m)}\|_\Omega}{\|G\|_\Omega}$	$\widehat{\eta}_G(\widehat{\lambda}_1,\widehat{v}_1)$	$\widehat{\eta}_G(\widehat{\lambda}_2,\widehat{v}_2)$	$\widehat{\eta}_G(\widehat{\lambda}_3,\widehat{v}_3)$	$\widehat{\eta}_G(\widehat{\lambda}_4,\widehat{v}_4)$	$\widehat{\eta}_G(\widehat{\lambda}_5,\widehat{v}_5)$	$\widehat{\eta}_G(\widehat{\lambda}_6,\widehat{v}_6)$
14	3.3e-5	2.5e-8	2.6e-8	7.0e-7	1.6e-6	1.7e-6	1.2e-6
18	1.8e-7	1.4e-10	1.4e-10	2.0e-9	2.9e-9	4.8e-9	2.0e-9
22	3.6e-10	9.1e-14	9.1e-14	2.7e-12	7.8e-13	3.7e-12	3.2e-12
28	1.5e-14	1.4e-15	1.4e-15	3.4e-15	7.4e-16	1.5e-15	4.9e-15

of Table 4.1. The computed eigenpairs $(\widehat{\lambda}_j, \widehat{v}_j)$ of $R^{(m)}(z)$, with $\widehat{\lambda}_j \in \Omega$, become the approximate eigenpairs of $G(z)$. We displayed the backward errors relative to $G(z)$ $\widehat{\eta}_G(\widehat{\lambda}_j, \widehat{v}_j)$ in Table 4.1, making sure that that $\widehat{\eta}_{R^{(m)}}(\widehat{\lambda}_j, \widehat{v}_j) \leqslant 7 \times 10^{-15}$. There we see that $\widehat{\eta}_G(\widehat{\lambda}_j, \widehat{v}_j) \leqslant \varepsilon$, with $\varepsilon = \|G - R^{(m)}\|_\Sigma / \|G\|_\Sigma$, and the backward errors decrease as ε decreases, as predicted by (4.6). Given that we know the exact eigenvalues, we compute the absolute error for the double and defective eigenvalue in 0, and the relative errors for the other eigenvalues, and we write them down in Table 4.2. Since in its first-order approximation the forward error is bounded from above by the product of the condition number times the backward error, we anticipate that the nonzero simple eigenvalues $\lambda_3, \dots, \lambda_6$ will have a condition number of order 10^3. A normwise condition number for a nonzero simple eigenvalue $\lambda \in \Omega$ of $G(z)$ with eigenvector v, which is consistent with the backward error defined in (2.18) is

$$\kappa_G(\lambda) = \limsup_{\varepsilon \to 0} \left\{ \frac{|\Delta\lambda|}{\varepsilon|\lambda|} : (G(\lambda + \Delta\lambda) + \Delta G(\lambda + \Delta\lambda))(v + \Delta v) = 0, \|\Delta G\|_\Omega \leqslant \varepsilon \|G\|_\Omega \right\}.$$

Following the proof of [GT17, Thm. 2.20] we get that

$$\kappa_G(\lambda) = \frac{\|G\|_\Omega \|w\|_2 \|v\|_2}{|\lambda||w^* G'(\lambda) v|},$$

were w is a left eigenvector of $G(z)$ with corresponding eigenvalue λ. Now for $G(z)$ in (4.8), we have $\|G\|_\Omega \approx \mathrm{e}^9 \approx 8 \times 10^3$. Also, all the left and right eigenvectors are nonzero multiples of $[1, \ -1]^T$. It is easy to see that

$$\|w\|_2 \|v\|_2 / (|\lambda_j| |w^* G'(\lambda_j) v|) = (2\pi)^{-1}$$

so that $\kappa_G(\lambda_j) \approx 1.3 \times 10^3$, $j = 3, \dots, 6$ as anticipated from the numerical experiments.

Table 4.2: Absolute and relative errors for approximate eigenvalues of $G(z)$ in (4.8) computed as eigenvalues of $R^{(m)}(z)$.

m	$\frac{\|G-R^{(m)}\|_\Omega}{\|G\|_\Omega}$	$\left\|\lambda_1 - \widehat{\lambda}_1\right\|$	$\left\|\lambda_2 - \widehat{\lambda}_2\right\|$	$\frac{\|\lambda_3-\widehat{\lambda}_3\|}{\|\lambda_3\|}$	$\frac{\|\lambda_4-\widehat{\lambda}_4\|}{\|\lambda_4\|}$	$\frac{\|\lambda_5-\widehat{\lambda}_5\|}{\|\lambda_5\|}$	$\frac{\|\lambda_6-\widehat{\lambda}_6\|}{\|\lambda_6\|}$
14	3.3e-5	1.7e-2	1.7e-2	6.4e-4	1.5e-3	1.5e-3	1.1e-3
18	1.8e-7	1.2e-3	1.3e-3	1.8e-6	2.7e-6	4.4e-6	1.8e-6
22	3.6e-10	3.2e-5	3.2e-5	2.5e-9	7.1e-10	3.4e-9	2.9e-9
28	1.5e-14	4.0e-6	4.0e-6	3.1e-12	6.8e-13	1.3e-12	4.5e-12

4.2 A BRIEF OVERVIEW OF CURRENT RATIONAL APPROXIMATION TECHNIQUES

In this section we give a brief overview of the approximation algorithms that influenced this chapter the most.

4.2.1 NLEIGS

The *fully rational Krylov method for nonlinear eigenvalue problems* (NLEIGS) is an algorithm proposed in [Güt+14] to solve nonlinear eigenvalue problems. It builds a rational approximant $R^{(m)}(z)$ in the form of Equation (4.1) and then returns its eigenvalues through a rational Krylov method. Here we are more interested in the construction of $R^{(m)}(z)$, but before going through the details, we need some background knowledge.

NLEIGS is a linearized rational interpolant, i.e., the sets of sampling points Σ and poles Ξ are prescribed a priori. Linearized rational interpolants allow us to build asymptotically optimal rational approximations, however a deep theoretical dive on how this is possible goes beyond the scope of this thesis, therefore we direct the interested reader to [Güt13]. Now assume for a moment that Σ, Ξ are general compact sets in $\overline{\mathbb{C}}$ at a positive distance one from the other. Then the pair (Σ, Ξ) is called *condenser* and related to it there is a positive real number $\text{cap}(\Sigma, \Xi)$ named *condenser capacity* [Bag67]. Computing this capacity for a given condenser is not a trivial task:

we direct to [Güt13] and [GT17] for some examples. Now we denote a *rational nodal function* by

$$s_m(z) = \prod_{j=0}^{m}(z-\sigma_j)/\prod_{j=1}^{m}(z-\xi_j),$$

where $\sigma_j \in \Sigma$ and $\xi_j \in \Xi$. As proved in [LS94], any sequence of nodal functions is bounded from below by

$$\limsup_{m\to\infty}\left(\frac{\sup_{z\in\Sigma} s_m(z)}{\inf_{z\in\Xi} s_m(z)}\right)^{1/m} \geqslant \mathrm{e}^{-1/\operatorname{cap}(\Sigma,\Gamma)}. \tag{4.9}$$

Finding a sequence of nodal functions $(s_m(z))_{m\in\mathbb{N}}$ such that the equality in (4.9) holds is called the *generalized Zolotarev problem*, because it becomes the third Zolotarev problem when the condenser is formed by real intervals [Güt13]. In 2006 Levi and Saff showed that the *generalized Leja–Bagby points* returns a sequence of optimal rational functions for the Zolotarev problem [Bag69], [LS06]. We obtain these points with the following greedy algorithm: we start with an arbitrary point $\sigma_0 \in \Sigma$, and then we recursively define σ_j and ξ_j as

$$\sigma_{j+1} := \arg\max_{z\in\Sigma}|s_j(z)|, \qquad \xi_{j+1} := \arg\min_{z\in\Gamma}|s_j(z)|. \tag{4.10}$$

We can explain the intuition behind the Leja–Bagby points in a particular case. Assume that $\Xi = \Gamma$ is now a closed contour that encloses Σ. Then the Walsh–Hermite formula tells us that

$$R^{(m)}(z) := \frac{1}{2\pi\mathrm{i}}\int_\Gamma\left(1-\frac{s_m(z)}{s_m(\zeta)}\right)\frac{G(\zeta)}{\zeta-z}\,d\zeta$$

is the unique rational matrix-valued function of type (m,m) that interpolates $G(z)$ at the nodes σ_j counting their multiplicities [Wal35]. The approximation error reads

$$\begin{aligned}\left\|G(z)-R^{(m)}(z)\right\|_2 &= \left\|\frac{1}{2\pi\mathrm{i}}\int_\Gamma\frac{s_m(z)}{s_m(\zeta)}\frac{G(\zeta)}{\zeta-z}\,d\zeta\right\|_2\\ &\leqslant K(G,\Omega,\Gamma)\frac{|s_m(z)|}{\min_{\zeta\in\Gamma}|s_m(\Gamma)|},\end{aligned} \tag{4.11}$$

where $K(G, \Omega, \Gamma) > 0$ is a constant that depends only on G, Ω, and Γ. Therefore, Equations (4.11) and (4.9) lead to

$$\limsup_{m\to\infty} \left\| G - R^{(m)} \right\|_{\Sigma}^{1/m} \leqslant \mathrm{e}^{-1/\operatorname{cap}(\Sigma,\Gamma)},$$

which means that choosing the Leja–Bagby points guarantees an exponential decay of the approximation error.

In order to practically construct $R^{(m)}(z)$ as in (4.1), we need to choose a series of basis functions $(b_j(z))_{j\in\mathbb{N}}$. NLEIGS opts for the degree-graded *rational Newton basis functions*, which are recursively defined as

$$b_0(z) = \frac{1}{\beta_0}, \qquad b_{j+1}(z) = \frac{z - \sigma_j}{\beta_{j+1}(1 - z/\xi_{j+1})} b_j(z), \tag{4.12}$$

where $(\sigma_j)_{j=0}^m \subset \Sigma$, $(\xi_j)_{j=1}^m \subset \Xi$, and β_j are nonzero parameters such that $\left\| b_j \right\|_{\Sigma} = 1$.

Remark 4.1. The silent hypothesis $\xi_j \neq 0$ does not affect the generality of the method, because if $G(z)$ had some poles in the proximity of the origin, we can simply consider a shifted version.

Finally, the computation of the constant coefficients R_j of (4.1) follows from the interpolation conditions $G(\sigma_j) = R^{(m)}(\sigma_j)$. When all the sampling points are distinct, we have $R_0 = \beta_0 G(\sigma_0)$, while (4.12) leads to

$$R_j = \frac{G(\sigma_j) - \sum_{k=1}^{j-1} b_k(\sigma_j) R_k}{b_j(\sigma_j)} = \frac{G(\sigma_j) - R^{(j-1)}(\sigma_j)}{b_j(\sigma_j)}, \qquad j = 1, \dots, m. \tag{4.13}$$

In Algorithm 4.1 we wrote down the pseudocode for the first part of NLEIGS. Güttel et al. [Güt+14], [EG19] truncate the approximation at step m when

$$\|R_m\|_F \leqslant \varepsilon \|R_0\|_F. \tag{4.14}$$

Indeed, if the rational approximation is converging, then

$$\left\| G - R^{(m)} \right\|_{\Sigma} = \max_{z\in\Sigma} \left\| \sum_{k=0}^{\infty} b_k(z) R_k - \sum_{k=0}^{m} b_k(z) R_k \right\|_2 \leqslant \sum_{k=m+1}^{\infty} \max_{z\in\Sigma} \|b_k(z)\|_2 \|R_k\|_2.$$

Algorithm 4.1: Pseudocode for the NLEIGS rational approximation.

```
   Input: $G, \Sigma, \Xi, m_{\max}, \varepsilon$
   Output: $R^{(m)}(z)$.
 1 $\sigma_0 \leftarrow$ random point in $\Sigma$
 2 $R_0 \leftarrow G(\sigma_0)$
 3 $b_0(z) \leftarrow 1$, $\beta_0 \leftarrow 1$
 4 $s_0(z) \leftarrow z - \sigma_0$, $j \leftarrow 0$
 5 while $\|R_j\|_F > \varepsilon \max_{0 \leqslant k \leqslant j} \|G(\sigma_k)\|_F/3$, and $j < m_{max}$ do
 6     $\sigma_{j+1} \leftarrow \arg\max_{z \in \Sigma} |s_j(z)|$
 7     $\xi_{j+1} \leftarrow \arg\min_{z \in \Xi} |s_j(z)|$
 8     $b_{j+1}(z) \leftarrow b_j(z)(z - \sigma_j)/(1 - z/\xi_{j+1})$
 9     $\beta_{j+1}(z) \leftarrow \max_{z \in \Sigma} |b_j(z)|$
10     $b_{j+1}(z) \leftarrow b_{j+1}(z)/\beta_{j+1}$
11     Build $R_{j+1}$ with Eq. (4.13)
12     $s_{j+1}(z) \leftarrow s_j(z)(z - \sigma_{j+1})/(z - \xi_{j+1})$
13     $j \leftarrow j + 1$
14 return $R^{(m)}(z)$
```

By construction, $\|b_k\|_\Sigma = 1$, thus

$$\left\|G - R^{(m)}\right\|_\Sigma \leqslant \sum_{j=m+1}^{\infty} \|R_j\|_2.$$

In addition, $\|G\|_\Sigma \geqslant n^{-1/2} \max_{0 \leqslant k \leqslant m} \|G(\sigma_k)\|_F$ and since we are supposing we are in the convergence regime, for m large enough it holds $\|R_j\|_F < \|R_m\|_F$ for $j > m$ and $\sum_{j=m+1}^{\infty} \|R_j\|_2 \leqslant \kappa \|R_m\|_F$ for some constant $\kappa > 1$ (in our implementation, we use $\kappa = 3$). Hence, instead of (4.14) we suggest to stop when

$$\|R_m\|_F \leqslant \frac{\varepsilon}{\kappa} \max_{0 \leqslant k \leqslant m} \|G(\sigma_k)\|_F, \tag{4.15}$$

which, once convergence has taken place, guarantees $\|G - R^{(m)}\|_\Sigma \leqslant \varepsilon \|G\|_\Sigma$.

Remark 4.2. Equation (4.15) is less strict than (4.14), given that $R_0 = G(\sigma_0)$. Therefore, NLEIGS equipped with (4.15) as stopping criterion returns a rational approximant with the required accuracy, but with a smaller degree m than when (4.14) is used.

If, on one hand, using the Leja–Bagby points leads to an exponential decay of the approximation error, on the other hand it requires a good knowledge of the singularities of the matrix-valued function $G(z)$. In fact, the final approximation cannot be good if the set Ξ is far away from the poles of $G(z)$. Nevertheless, there exist instances where this knowledge is unavailable. For example, $G(z)$ may be in black-box form, or understanding where the singularities lie from the split form could be too difficult; finally, the end-users may be more interested in a simpler algorithm, where they do not need to specify many parameters. For these reasons, algorithms where the poles do not need to be specified a priori are always welcomed by the community. Among them, in recent years the *Adaptive Antoulas–Anderson (AAA)* method has been having great success, and thus it is the topic of the next section.

4.2.2 The AAA algorithm

In 2018 Nakatsukasa, Sète, and Trefethen proposed the *Adaptive Antoulas–Anderson (AAA)* for the approximation of scalar functions [NST18]. It immediately surged in popularity and several authors generalised it to matrix-valued functions $G(z)$ [Hoc17; Lie+18].

The original AAA algorithm aims to approximate a scalar function $g(z)\colon \Omega \to \mathbb{C}$ with a sequence of rational functions

$$r^{(m)}(z) = \frac{\displaystyle\sum_{i=1}^{m} \frac{g(\sigma_i)w_i}{z-\sigma_i}}{\displaystyle\sum_{i=1}^{m} \frac{w_i}{z-\sigma_i}} = n^{(m)}(z)/d^{(m)}(z), \tag{4.16}$$

where $w_i \in \mathbb{C}$ are weights, and $(\sigma_i)_{i=1}^{m} \subset \Sigma \subset \Omega$ are nested sequences of sample/interpolation points. Equation (4.16) is often called the *barycentric representation of* $r^{(m)}$. We summarise its properties in the next theorem [NST18].

Theorem 4.1. *Consider the arbitrary distinct complex sample points $\sigma_1, \dots, \sigma_m$. As the complex values $g_1 := g(\sigma_1), \dots g_m := g(\sigma_m)$ range over all complex values and the weights $w_1, \dots, w_m$ range over all the nonzero complex values, the functions $r^{(m)}(z)$ in (4.16) range all over the set of the $(m-1, m-1)$ rational functions that have no poles at the points σ_j. Moreover, $r(\sigma_j) = g_j$ for each j.*

The core of the AAA procedure is a greedy selection of the points σ_i, one at a time, from the set $\Sigma := \{\sigma_1, \dots, \sigma_M\}$, where we assume $M \gg 1$. We summarise the procedure in the upcoming paragraphs and in Algorithm 4.2. More precisely, at every step m we will have an approximation $r^{(m)}(z)$ on the support points $\{\sigma_1, \dots, \sigma_m\}$ that minimizes the 2-norm on

$$\Sigma^{(m)} = \Sigma \backslash \{\sigma_1, \dots, \sigma_m\}, \tag{4.17}$$

i.e., the points that are not yet the support ones. The goal is approximating

$$g(z) \approx r^{(m)}(z), \qquad z \in \Sigma,$$

or equivalently

$$g(z) d^{(m)}(z) \approx n^{(m)}(z), \qquad z \in \Sigma^{(m)}, \tag{4.18}$$

where we substituted Σ with $\Sigma^{(m)}$, due to the presence of the poles of $n^{(m)}(z)$ and $d^{(m)}(z)$. Now assume we have just completed the $(m-1)$-th iteration. Then the next support point σ_m is chosen so that

$$\max_{\sigma \in \Sigma^{(m-1)}} \left| g(\sigma) - r^{(m-1)}(\sigma) \right| = \left| g(\sigma_m) - r^{(m-1)}(\sigma_m) \right|, \tag{4.19}$$

i.e., it is the point that maximises the residual on $\Sigma^{(m-1)}$. More precisely, we write $\Sigma^{(m)} = [\sigma_1^{(m)}, \dots, \sigma_{M-m}^{(m)}]^T$ as a vector and we let $G^{(m)} = [g_1^{(m)}, \dots, g_{M-m}^{(m)}]^T$ be the vector of the evaluations $g(\Sigma^{(m)})$. We are looking for a normalised vector (the weights)

$$w = \begin{bmatrix} w_1 & \dots & w_m \end{bmatrix}^T, \qquad \|w\| = 1,$$

that minimizes the 2-norm of

$$\sum_{j=1}^{m} \frac{w_j (g_i^{(m)} - g_j)}{\sigma_i^{(m)} - \sigma_j} \qquad i = 1, \dots, M - m. \tag{4.20}$$

Equation (4.20) can be rewritten in matrix form as

$$\min_{\|w\|=1} \left\| A^{(m)} w \right\|_2,$$

where

$$A^{(m)} = \begin{bmatrix} \frac{g_1^{(m)} - g(\sigma_1)}{\sigma_1^{(m)} - \sigma_1} & \cdots & \frac{g_1^{(m)} - g(\sigma_m)}{\sigma_1^{(m)} - \sigma_m} \\ \vdots & \ddots & \vdots \\ \frac{g_{M-m}^{(m)} - g(\sigma_1)}{\sigma_{M-m}^{(m)} - \sigma_1} & \cdots & \frac{g_{M-m}^{(m)} - g(\sigma_m)}{\sigma_{M-m}^{(m)} - \sigma_m} \end{bmatrix} \in \mathbb{C}^{(M-m)\times m}. \tag{4.21}$$

Finally, the vector of weights w is the final right singular vector of the reduced SVD $A^{(m)} = U\Sigma V^*$. The procedure then stops when

$$\left\| g - r^{(m)} \right\|_\Sigma = \left\| g - r^{(m)} \right\|_{\Sigma^{(m)}} \leqslant \varepsilon \|g\|_\Sigma, \tag{4.22}$$

for a given ε, where the norm on Σ defined in (4.3) reduces to $\|g\|_\Sigma = \max_{z\in\Sigma} |g(z)|$ for a scalar function $g(z)$.

Remark 4.3. The AAA approximant can stagnate if the tolerance ε is too small. If that is the case, then *numerical Froissart doublets* appear [Fro69]. These are poles with very small residues or pairs of poles and support points that are so close together that they almost cancel. In order to remove them, we first identify the spurious poles thanks to their residues being smaller than a given threshold ε, say 10^{-13}; then, we remove the nearest support point from the set of support points, and finally we compute a new SVD to solve the least-squares problem (4.20). We direct the reader to [NST18, Section 5] for further details and references on this topic.

Remark 4.4. As opposed to NLEIGS, the greedy nature of the AAA algorithm and its variants, it is able to approximate functions that are *not* holomorphic in Ω. Given that in later sections we will use it as well as a first step of another approximation, we will focus only on holomorphic functions. Nonetheless, in the numerical examples we will provide examples of functions without this smoothness property.

Algorithm 4.2: Pseudocode for the AAA algorithm

```
Input: $g, \Sigma, m_{\max}, \varepsilon$
Output: $r(z)$.
1  $\Sigma^{(0)} \leftarrow \Sigma$
2  $r^{(0)}(z) \leftarrow 0$, $n^{(0)}(z) \leftarrow 0$, $d^{(0)}(z) \leftarrow 1$
3  $m \leftarrow 1$
4  while $m \leqslant m_{max}$ do
5      Compute $\sigma_m = \arg\max_{z \in \Sigma^{(m-1)}} \left| g(z) d^{(m-1)}(z) - n^{(m-1)}(z) \right|$
6      $\Sigma^{(m)} \leftarrow \Sigma^{(m-1)} \backslash \{\sigma_m\}$
7      Compute the SVD of matrix $A^{(m)}$ (4.21)
8      Retrieve the vector of weights $w$
9      Build $r^{(m)}(z)$
10     if $\left\| g - r^{(m-1)} \right\|_\Sigma < \varepsilon \|g\|_\Sigma$ or $m = m_{max}$ then
11         return $r^{(m)}(z)$
12     $m \leftarrow m + 1$
```

The most appreciated feature of AAA is its speed: the expensive part is the computation of the $(M-j) \times j$ SVDs, with $j = 1, \dots m$, therefore the cost is $O(m^3 M)$, where m is usually very small ($m < 50$) [NST18]. Finally, in some applications retrieving the zeros and the poles of $r^{(m)}(z)$ is fundamental, as we will show in Section 4.3. As opposed to other algorithms, these quantities are not readily available during the AAA procedure. Nevertheless, computing them is as easy as solving two linear eigenvalue problems:

$$\begin{bmatrix} 0 & w_1 & w_2 & \dots & w_m \\ 1 & \sigma_1 & & & \\ 1 & & \sigma_2 & & \\ \vdots & & & \ddots & \\ 1 & & & & \sigma_m \end{bmatrix} = \lambda \begin{bmatrix} 0 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix} \tag{4.23}$$

gives us the zeros of $d^{(m)}(z)$, i.e., the poles of $r^{(m)}(z)$, while the pencil where we substitute w_j with $g_j w_j$ returns the zeros of $n^{(m)}(z)$. This costs $O(m^3)$ as well, therefore it does not increase the asymptotical complexity of the algorithm.

4.2.3 The set-valued AAA and the weighted AAA

Since the first article on the AAA algorithm for scalar functions appeared, many researchers aimed to extend it to a set of multiple scalar functions, and thus to the matrix-valued case. Hochman [Hoc17], and Lietaert et al. [Lie+18] proposed the *fastAAA* and the *set-valued AAA* algorithms, respectively. They both require a split form of the matrix-valued function $G(z)$, and only few details differ from one and the other, thus we will treat them as a single work. The first naive idea is approximating each scalar function $g_j(z)$ of (4.4) with s calls of AAA on the set Σ. However, we will end up with s $r_i(z)$ functions whose support points are different subsets of Σ. For numerical stability, it is more beneficial to find common support points for all the functions at the same time. This leads to the rational approximants

$$r_j^{(m)}(z) = \sum_{i=0}^{m} \frac{g_j(\sigma_i) w_i}{z - \sigma_i} \Bigg/ \sum_{i=0}^{m} \frac{w_i}{z - \sigma_i}, \quad j = 1, \dots, s, \tag{4.24}$$

just like in (4.16). This time the m-th support point is chosen such that

$$\max_{i,j} \left| g_i(\sigma_j) - r^{(m-1)}(\sigma_j) \right|$$

attains its maximum. The core of set-valued AAA is the same of the original one for scalar functions, therefore we direct the interested reader to the cited references. We only desire to point out that the Loewner matrix $A^{(m)}$ of (4.21) becomes much taller:

$$A^{(m)} = \begin{bmatrix} \frac{g_{1,1}^{(m)} - g_1(\sigma_1)}{\sigma_1^{(m)} - \sigma_1} & \cdots & \frac{g_{1,1}^{(m)} - g_1(\sigma_m)}{\sigma_1^{(m)} - \sigma_m} \\ \vdots & \ddots & \vdots \\ \frac{g_{1,M-m}^{(m)} - g_1(\sigma_1)}{\sigma_{M-m}^{(m)} - \sigma_1} & \cdots & \frac{g_{1,M-m}^{(m)} - g_1(\sigma_m)}{\sigma_{M-m}^{(m)} - \sigma_m} \\ \frac{g_{2,1}^{(m)} - g_2(\sigma_1)}{\sigma_1^{(m)} - \sigma_1} & \cdots & \frac{g_{2,1}^{(m)} - g_2(\sigma_m)}{\sigma_1^{(m)} - \sigma_m} \\ \vdots & \ddots & \vdots \\ \frac{g_{2,M-m}^{(m)} - g_2(\sigma_1)}{\sigma_{M-m}^{(m)} - \sigma_1} & \cdots & \frac{g_{2,M-m}^{(m)} - g_2(\sigma_m)}{\sigma_{M-m}^{(m)} - \sigma_m} \\ \vdots & \vdots & \vdots \\ \frac{g_{s,1}^{(m)} - g_s(\sigma_1)}{\sigma_1^{(m)} - \sigma_1} & \cdots & \frac{g_{s,1}^{(m)} - g_s(\sigma_m)}{\sigma_1^{(m)} - \sigma_m} \\ \vdots & \ddots & \vdots \\ \frac{g_{s,M-m}^{(m)} - g_s(\sigma_1)}{\sigma_{M-m}^{(m)} - \sigma_1} & \cdots & \frac{g_{s,M-m}^{(m)} - g_s(\sigma_m)}{\sigma_{M-m}^{(m)} - \sigma_m} \end{bmatrix} \in \mathbb{C}^{(M-m)s \times m} \tag{4.25}$$

It follows that each step of the procedure may become quite expensive when s is large. Therefore they proposed a fast method to solve the least-squares problem (4.20) [Lie+18, Section 2.3]. We write $A^{(m-1)} = QH$, where $Q \in \mathbb{C}^{(M-m+1)\times(m-1)}$ is orthonormal, and the right singular vectors of $A^{(m-1)}$ are the same of the matrix $H \in \mathbb{C}^{(m-1)\times(m-1)}$. At the m-th step, we can update Q by adding the last column of $A^{(m)}$, by removing the s rows corresponding to the support point σ_m, and by reorthogonalizing this new matrix. This can be done cheaply: define $Q_s \in \mathbb{C}^{s\times m}$ the matrix whose rows are the rows removed from Q, and let $\widetilde{Q}$ the matrix obtained from Q after the removal of those s rows. The orthogonality of Q yields

$$\widetilde{Q}^*\widetilde{Q} = I_m - Q_s^* Q_s = S^*S,$$

where S^*S is the Cholesky decomposition. It follows that $\widetilde{Q}S^{-1}$ is orthogonal and $(\widetilde{Q}, H) \leftarrow (\widetilde{Q}S^{-1}, SH)$ is the desired update. Unfortunately, this fast reorthogonalization may fail at later steps of the algorithm when the threshold ε is small if there is some stagnation. It is widely known that the Cholesky factorization is stable (see, for example, [GV96; Hig90; Wil68] and the citations therein), nevertheless under these circumstances the matrix Q_s is nearly orthogonal, thus $I_m - Q_s^* Q_s$ may become numerically positive semi-definite. When this happens, we have to go back to computing the SVD of $A^{(m)}$ from scratch.

In the original set-valued algorithm the stopping criterion that follows from 4.22 is

$$\max_{1\leqslant j\leqslant s} \left\| g_j - r_j^{(m)} \right\|_\Sigma \leqslant \varepsilon \left\| g_j \right\|_\Sigma. \tag{4.26}$$

It springs to the eye that nowhere in (4.26) the matrix coefficients A_j appear, which seems suboptimal. Intuitively, if $\|A_j\|_2$ is much smaller than the other coefficients, then the approximant $r_j^{(m)}(z)$ does not need to be very precise. Indeed, following the analysis of Section 4.1, we get

$$\left\| G - R^{(m)} \right\|_\Sigma = \max_{z\in\Sigma} \left\| \sum_{j=1}^{s} (g_j(z) - r_j^{(m)}(z)) A_j \right\|_2 \leqslant \sum_{j=1}^{s} \left\| g_j - r_j^{(m)} \right\|_\Sigma \|A_j\|_2.$$

So we propose to use the stopping criterion

$$\sum_{j=1}^{s} \left\| g_j - r_j^{(m)} \right\|_{\Sigma} \|A_j\|_F \leqslant \varepsilon\beta, \tag{4.27}$$

where β is a lower bound on $\|G\|_{\Sigma}$ that we assume can be computed cheaply. Under the assumption that (4.27) holds, we have

$$\left\| G - R^{(m)} \right\|_{\Sigma} \leqslant \sum_{j=1}^{s} \left\| g_j - r_j^{(m)} \right\|_{\Sigma} \|A_j\|_2 \leqslant \sum_{j=1}^{s} \left\| g_j - r_j^{(m)} \right\|_{\Sigma} \|A_j\|_F \leqslant \varepsilon\beta \leqslant \varepsilon \|G\|_{\Sigma}.$$

It follows that if all $r_j^{(m)}(z)$s satisfy (4.27), then the rational approximant $R^{(m)}(z)$ satisfies (4.2). Since the stopping criterion weights differently each scalar function g_j, we named the AAA approximant $R^{(m)}(z)$ hereby obtained, the *weighted AAA rational approximant*. On the other hand, the original stopping criterion (4.26) leads to

$$\left\| G - R^{(m)} \right\|_{\Sigma} \leqslant \sum_{j=1}^{s} \left\| g_j - r_j^{(m)} \right\|_{\Sigma} \|A_j\|_2 \leqslant \left(s \max_{1 \leqslant j \leqslant s} \|g_j\|_{\Sigma} \|A_j\|_2 \right) \varepsilon.$$

Since $(s \max_{1\leqslant j\leqslant s} \|g_j\|_{\Sigma} \|A_j\|_2)/\|G\|_{\Sigma} \geqslant 1$, Equation (4.2) will not generally hold. Consequently, if the lower bound β on $\|G\|_{\Sigma}$ is sharp, the construction of $R^{(m)}(z)$ will stop earlier with (4.27) than with (4.26). We will illustrate this behaviour in the numerical experiments of Section 4.4. Finally, weighted AAA is scaling independent. This means it returns the same approximant when applied to the $G(z)$ in split form as in (4.4) and to $G(z) = \sum_{j=1}^{s} g_j(z) B_j$ with $g_j(z) = \alpha_j g_j(z)$ and $A_j = \alpha_j^{-1} B_j$, $\alpha_j \neq 0$. Also, it returns $\alpha R^{(m)}(z)$ when applied to $\alpha G(z)$, where $R^{(m)}(z)$ is the approximant to $G(z)$.

Remark 4.5. A cheap way to compute a lower bound of $\|G\|_{\Sigma}$ is the following. Consider a normally distributed vector $u \in \mathbb{C}^n$ with unit length, and let $u_j = A_j u$, for $j = 1, \dots, s$. Then we define

$$\beta := \max_{z\in\Sigma} \left\| \sum_{j=1}^{s} g_j(z) u_j \right\|_2 \leqslant \|G\|_{\Sigma}.$$

Note that an unlucky selection of u may lead to a lower bound of several orders of magnitude smaller than $\|G\|_{\Sigma}$. This poor lower bound will cause the algorithm to perform a few unnecessary steps, thus returning an approximant of degree larger than needed.

4.2.4 The Cauchy approximation and a brief wrap-up

In Chapter 2 we saw how to use the Cauchy's integral formula in Theorem 1.2 to solve a meromorphic eigenvalue problem on Ω. Saad et al. realised they could take advantage of the same result to return a rational approximant in the holomorphic case [EMS20]. Recalling that $g_j(z) \in \mathcal{H}(\Omega_0, \mathbb{C})$ and assuming that the boundary $\partial\Omega_0$ is smooth enough, Cauchy's formula yields

$$g_j(z) = \frac{1}{2\pi \mathrm{i}} \int_{\partial\Omega_0} \frac{g_j(u)}{u - z} du, \quad z \in \Omega_0 \backslash \partial\Omega_0, \quad j = 1, \dots, s.$$

If we build a parametrization $\gamma \colon [0, 2\pi] \to \partial\Omega_0$, the substitution $u = \gamma(t)$ yields

$$g_j(z) = \frac{1}{2\pi \mathrm{i}} \int_0^{2\pi} \frac{g_j(\gamma(t))\gamma'(t)}{\gamma(t) - z} dt.$$

Thus, if we employ a quadrature with $m + 1$ nodes ξ_i and weights ω_i, then we can build the rational approximant

$$r_j^{(m)}(z) = \sum_{k=0}^{m} \frac{w_k g_j(\xi_k)}{\xi_k - z}, \quad j = 1, \dots, s. \tag{4.28}$$

Due to its nature, this *Cauchy approximation* is not an interpolation technique: there are no points σ_k where we force $g_j(\sigma_k) = r_j^{(m)}(\sigma_k)$. Rather, it follows from (4.28) that the poles ξ_k of $r_j^{(m)}(z)$ must not be poles of $g_j(z)$.

The most important merit of this algorithm is the easiness of implementation of its core: if the shape of Ω is not too complicated, we can just take a slightly larger Ω_0 with a smooth contour, and a quadrature rule. For example, when Ω_0 is a disk, the trapezoidal rule converges exponentially [TW14, Theorem 2.2], thus the weights simply become $\omega_k = (m+1)^{-1}$. However, things are not so simple when we try to make it more robust. First of all, at runtime the user does not have a clear grasp of what will happen. Generally, most quadrature rules converge exponentially to the true integral, as we wrote more in depth in Section 3.4, but a quantitative analysis seems not possible: given $\varepsilon \ll 1$, the algorithm cannot automatically choose a degree m such that (4.2) holds. In addition, setting Ω_0, and thus $\partial\Omega_0$, is not obvious: if $\partial\Omega_0$ is too close to $\partial\Omega$, then at the points $\sigma_i \in \Omega$ close to the boundary the approximation will not be good, due to the quadrature points ξ_k being poles for $r_j^{(m)}(z)$, but not for $g_j(z)$; if it is too far, the behaviour of $g_j(z)$ on $\partial\Omega_0$ can be very different from the one in Ω, and this may compromise the quality of the approximation as well. The upcoming example clarifies these difficulties.

Example 4.2. *Let $G(z)$ be the 2-by-2 matrix-valued function in (4.8), where the target set Ω is $\mathcal{D}(0, \rho)$ and $\rho = 3$. We can rewrite $G(z)$ in split form as*

$$G(z) = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} + \mathrm{e}^{\mathrm{i}z^2} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}. \tag{4.29}$$

We build a Cauchy approximant $R^{(m)}(z)$ using the trapezoidal rule on the contours $\partial\Omega_0$, where $\Omega_0 = \mathcal{D}(0, \alpha\rho)$ and $\alpha \in \{1.05, 1.1, 1.25, 1.5\}$. We are interested in the error on Σ, which is formed by 400 random points in Ω and 100 uniformly distributed points on $\partial\Omega$. For each value of α we have plotted in Figure 4.1 the error $\|G - R^{(m)}\|_\Sigma / \|G\|_\Sigma$ as the number of quadrature points, i.e., the degree of $R^{(m)}(z)$, increases. On one hand, we witness that the convergence is very slow when α is close to 1. This is an expected behaviour, due to the presence of the poles, which causes the approximation to be bad on $\partial\Omega$ and on the points in Σ near $\partial\Omega$. On the other hand, the rate of convergence increases with α, because we are distancing the poles ξ_i of $R^{(m)}$ (the quadrature points) from Ω, but the limiting accuracy increases as well, because the values the function $G(z)$ assumes on $\partial\Omega_0$ are much larger than the values in Ω.

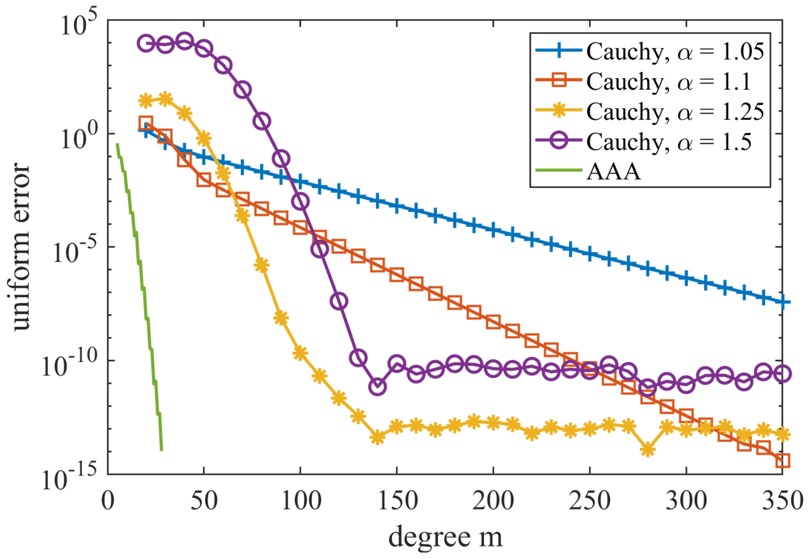


Figure 4.1: Demonstration of four different choices for the contour $\partial\Omega_0$ (a circle of centre 0 and radius $\alpha\rho$) for the Cauchy approximation of G in (4.8) on Ω, the disk of centre 0 and radius $\rho = 3$.

4.3 A TWO-PHASE ALGORITHM FOR BLACK-BOX FUNCTIONS

In Section 4.2 we reviewed some approximation algorithms and we proposed an improvement for the original set-valued AAA. However, all these algorithms work when the function $G(z)$ is given in split form (4.4). If $G(z)$ is in black box form, we need to adopt other strategies.

In 2018 Elsworth and Güttel proposed another generalisation to the AAA algorithm for black-box functions [EG19]. First of all, they draw two normally distributed random vectors $u, v \in \mathbb{C}^n$ of unit length and build the surrogate scalar function

$$g(z) = u^* G(z) v. \tag{4.30}$$

Then they approximate $g(z)$ through the AAA algorithm of section 4.2.2 and return the approximant $r^{(m)}(z)$ such that (4.22) holds. They argue that the region of analiticity of $g(z)$ must be similar to $G(z)$, therefore the support points σ_j and the weights

w_j of $r^{(m)}(z)$ should be a good choice for a rational approximation of $G(z)$. Thus they define

$$R^{(m)}(z) = \sum_{j=0}^{m} \frac{G(\sigma_j)w_j}{z-\sigma_j} \Bigg/ \sum_{j=0}^{m} \frac{w_j}{z-\sigma_j}. \tag{4.31}$$

Finally, they showed how to rewrite the barycentric representation (4.31) in the Newton basis of (4.1). They named this algorithm *surrogate AAA*.

Unfortunately, there is no guarantee that $R^{(m)}(z)$ approximates $G(z)$ in Σ and thus in Ω. Clearly, in the edge scenario where

$$G(z) = \begin{bmatrix} g_{11}(z) & 0 \\ 0 & 1 \end{bmatrix}, \qquad u = v = e_2,$$

surrogate AAA will fail at producing any reasonable result. Nevertheless we can set it as a "preliminary step" in a two-phase approximation algorithm. The upcoming sections are dedicated to the exploration of several possibilities.

Finally, we recall that our goal is returning a rational approximant $R^{(m)}(z)$ such that the relative error is less than a given threshold ε, as written in (4.2). However, as we showed in Section 4.2.1, computing $\|G\|_\Sigma$ can be difficult, therefore we settle for a lower bound β. When the surrogate AAA is used, a good lower bound is readily available. In fact, the surrogate function (4.30) can be seen as $g(z) = u^*\widetilde{g}(z)$, where $\widetilde{g}(z) = G(z)v$ is evaluated for all $z \in \Sigma$. Thus we define

$$\widetilde{\beta} = \max_{z\in\Sigma} \|\widetilde{g}(z)\|_2 \leqslant \|G\|_\Sigma. \tag{4.32}$$

4.3.1 Surrogate AAA with exact or relaxed search

In the previous paragraphs we have seen that the "original sin" of surrogate AAA lies in only considering a one-dimensional projection of $G(z)$, instead of it as a whole.

Therefore, the first refinement we considered was changing the support points drawn from Σ. More precisely, assume that we have the rational approximant

$$R^{(d)}(z) = \sum_{i=0}^{d} \frac{G(\sigma_i)w_i}{z-\sigma_i} \Big/ \sum_{i=0}^{d} \frac{w_i}{z-\sigma_i} \tag{4.33}$$

obtained after d steps of the surrogate AAA algorithm. Then we modify the cycle as follows. Instead of computing σ_{d+1} as in (4.19), it is determined as

$$\sigma_k = \underset{z\in\Sigma^{(k-1)}}{\arg\max} \left\| G(z) - R^{(k-1)}(z) \right\|_F, \quad k > d, \tag{4.34}$$

where $\Sigma^{(k-1)}$ is defined in (4.17). The weights w_i are computed as for the surrogate AAA approximation. The procedure stops at step $k = m$ when

$$\max_{z\in\Sigma^{(m)}} \left\| G(z) - R^{(m)}(z) \right\|_F \leqslant \varepsilon\beta \tag{4.35}$$

holds, where $\beta := \max\{\widetilde{\beta}, \max_{z\in\Sigma^{(m)}} \|G\|\}$ is the lower bound for $\|G\|_\Sigma$ in (4.32). It follows that this rational approximant $R^{(m)}(z)$ satisfies (4.2). We named this approach *surrogate AAA with exact search on* Σ.

Finding σ_k in (4.34) is expensive, because we have to compute the Frobenius norm of a $n \times n$ matrix at $M - k$ points. We suggest two ways to reduce the complexity.

1. Since $G(z)$ is holomorphic in Ω, we can remove all the sampling points in the interior part of Σ and only work on the boundary $\partial\Sigma := \Sigma \cap \partial\Omega$, which we call *surrogate AAA with exact search on* $\partial\Sigma$.

2. Instead of choosing σ_k as in (4.34), we shuffle the points in $\Sigma^{(k)}$ and take the first support point $\sigma_k \in \Sigma^{(k)}$ such that

$$\left\| G(\sigma_k) - R^{(k-1)}(\sigma_k) \right\|_F > \varepsilon\widetilde{\beta}. \tag{4.36}$$

   We refer to this approach as *surrogate AAA with relaxed search*. We point out that the shuffle is necessary, because otherwise the approximation would have

Algorithm 4.3: Pseudocode for the SURROGATE AAA WITH EXACT SEARCH algorithm.

```
Input: $G, \Sigma, m_{\max}, \varepsilon$
Output: $R^{(m)}(z)$.
1  Draw random vectors $u, v \in \mathbb{C}^n$
2  Build surrogate function $g(z) = u^* G(z) v$
3  $r^{(0)}(z) \leftarrow 0$, $n^{(0)}(z) \leftarrow 0$, $d^{(0)}(z) \leftarrow 1$
4  $\Sigma^{(0)} \leftarrow \Sigma$, $d \leftarrow 1$
5  while $d \leqslant m_{max}$ do
6      Compute $\sigma_d = \arg\max_{z \in \Sigma^{(d-1)}} \left| g(z) d^{(d-1)}(z) - n^{(d-1)}(z) \right|$
7      $\Sigma^{(d)} \leftarrow \Sigma^{(d-1)} \backslash \{\sigma_d\}$
8      Compute the SVD of matrix $A^{(d)}$ (4.21)
9      Retrieve the vector of weights $w$
10     Build $r^{(d)}(z)$
11     $d \leftarrow d + 1$
12     if $\left\| g - r^{(d-1)} \right\|_\Sigma < \varepsilon \|g\|_\Sigma$ or $d = m_{max} + 1$ then
13         continue
14 Build $R^{(d-1)}(z)$, $m \leftarrow d$
15 while $m \leqslant m_{max}$ do
16     Compute $\sigma_m = \arg\max_{z \in \Sigma^{(m-1)}} \left\| G(z) - R^{(m)}(z) \right\|_F$
17     $\Sigma^{(m)} \leftarrow \Sigma^{(m-1)} \backslash \{\sigma_m\}$
18     Compute the SVD of matrix $A^{(m)}$ (4.21)
19     Retrieve the vector of weights $w$
20     Build $R^{(m)}(z)$
21     if $\left\| G - R^{(m)} \right\|_\Sigma > \varepsilon \|G\|_\Sigma$ or $m = m_{max}$ then
22         return $R^{(m)}(z)$
23     $m \leftarrow m + 1$
```

a bias on the points that appear first in Σ. We call this approach *surrogate AAA with relaxed search.*

In Algorithm 4.3, we summarise the exact search variant on the entire set Σ. The only difference with the relaxed search lies at line 16, where we substitute the condition with (4.36). The numerical experiments in section 4.4 show that this refinement works well when the tolerance ε is not too strict. However, when ε is really small, computing the weights w_i through the surrogate function prevents the procedure to reach the desired accuracy.

4.3.2 NLEIGS with poles from surrogate AAA

In (4.23) we showed how the AAA algorithm can return the poles of the rational approximant $r^{(m)}(z)$. The NEP module inside the SLEPc library [CR19] builds a surrogate AAA approximant of the scalar function f in (4.30), say $r^{(d)}(z)$ of degree d, then retrieve its poles Ξ, and feed the condenser (Σ, Ξ) to the NLEIGS algorithm 4.1 summarised in Section 4.2.1.

This method combines the advantages of AAA and NLEIGS: on one hand, we have the convenience of the first algorithm, which only requires Σ as input; on the other, we have the robustness and parameter control given by the latter. More precisely, we mean that the set of poles Ξ can be preprocessed before being given as an input. For instance, we might have to remove unwanted poles inside Ω, if the AAA algorithm returns any. We called it *NLEIGS with poles from surrogate AAA algorithm*.

It may often occur that the number of poles $\xi_1, \ldots, \xi_d$ is not sufficient for NLEIGS to reach its stopping criterion (4.15), i.e., d is smaller than the optimal degree m of the approximant $R^{(m)}(z)$. When this is the case, the SLEPc module adds extra points at infinity, i.e., $\xi_{d+1} = \cdots = \infty$. We show that this approach is not ideal and fall back to a polynomial approximation. In fact, define the nodal

$$q_d(z) := (z - \xi_1) \cdots (z - \xi_d)$$

and the denominator polynomial

$$s_d(z) := (z - \sigma_0)(z - \sigma_1) \cdots (z - \sigma_d).$$

Then setting additional poles at infinity is equivalent to computing a polynomial interpolant $P(z)$ of degree $m - d - 1$ to the scaled error function

$$E(z) := q_d(z)(G(z) - R^{(d)}(z))/s_d(z) \tag{4.37}$$

and then having

$$R^{(m)}(z) := R^{(d)}(z) + s_d(z)P(z)/q_d(z). \tag{4.38}$$

We recall that a rational function $r(z) = p(z)/q_d(z)$ of type (m, m) with a fixed denominator is uniquely determined by $m + 1$ interpolation conditions $r(\sigma_j) = g_j$ at distinct points σ_j, with the caveat that $r(\sigma_j)$ is well defined for every j. Now, the same holds true in the matrix-valued case, where $R(z) = P(z)/q_d(z)$ is the interpolant and $P(z)$ is a matrix polynomial of degree m, because each entry is a scalar rational function with the interpolation property.

At the end of the surrogate AAA step, we have $R^{(d)}(z) = N(z)/q_d(z)$, for a given matrix polynomial $N(z)$; it interpolates $G(z)$ at the nodes $\sigma_0, \dots, \sigma_d$ by construction. After $m - d$ Leja–Bagby steps with poles $\xi_{d+1} = \cdots = \xi_m = \infty$ and nodes $\sigma_{d+1}, \dots, \sigma_m$, we get a rational interpolant $R^{(m)}(z) = \widetilde{N}(z)/q_d(z)$, which satisfies the $m + 1$ conditions at $\sigma_0, \dots, \sigma_m$. In addition, it has the same denominator of (4.38), thus it must coincide with it by the uniqueness of the interpolant.

The above paragraphs show that setting poles at infinity implies reverting to the polynomial interpolation of (4.37). The convergence of this second step is governed by the analyticity region of $E(z)$ in the neighbourhood of Ω. We point out that the σ_j are not poles of $E(z)$ due to the interpolation conditions, and $q_d(z)R^{(d)}(z)$ is a matrix polynomial, therefore this region coincides with the analyticity region of $G(z)$.

Falling back to the polynomial interpolation might cause a severe slowdown in the convergence of the algorithm. At the same time, the surrogate phase only returns d poles. Hence we propose to cyclically repeat these poles. More precisely, as soon as the number k of Leja–Bagby points generated by the algorithm exceeds d, the expression for the poles ξ_k in (4.10) becomes

$$\xi_k = \xi_{1+(k-1 \mod d)}. \tag{4.39}$$

We refer to this algorithm with the name *d-cyclic Leja–Bagby procedure*. Note that the surrogate AAA retrieves the poles ξ_j thanks to a linear eigenvalue problem, therefore

Algorithm 4.4: Pseudocode for NLEIGS WITH POLES FROM SURROGATE AAA algorithm. The poles Ξ are repeated cyclically if needed.

```
   Input: $G, \Sigma, m_{\max}, \varepsilon$
   Output: $R(z)$.
1  Draw random vectors $u, v \in \mathbb{C}^n$
2  Build surrogate function $g(z) = u^* G(z) v$
3  $r^{(0)}(z) \leftarrow 0$, $n^{(0)}(z) \leftarrow 0$, $d^{(0)}(z) \leftarrow 1$
4  $\Sigma^{(0)} \leftarrow \Sigma$, $d \leftarrow 1$
5  while $d \leqslant m_{max}$ do
6      Compute $\sigma_d = \arg\max_{z \in \Sigma^{(d-1)}} \left| g(z) d^{(d-1)}(z) - n^{(d-1)}(z) \right|$
7      $\Sigma^{(d)} \leftarrow \Sigma^{(d-1)} \backslash \{\sigma_d\}$
8      Compute the SVD of matrix $A^{(d)}$ (4.21)
9      Retrieve the vector of weights $w$
10     Build $r^{(d)}(z)$
11     $d \leftarrow d + 1$
12     if $\left\| g - r^{(d-1)} \right\|_\Sigma < \varepsilon \|g\|_\Sigma$ or $d = m_{max} + 1$ then
13         continue
14 Compute the poles $\widetilde{\Xi}$ from (4.23)
15 $\Xi \leftarrow$ sorted $\widetilde{\Xi}$ with (4.10)
16 return NLEIGS($G, \Sigma, \Xi, m_{\max}, \varepsilon$) (Algorithm 4.1)
```

they do not have the natural order that the Leja–Bagby sampling would have gotten if it were used from the beginning. Therefore, before calling the d-cyclic Leja–Bagby procedure, we need to reorder the poles. This can be done by defining once more the nodal function $s_0(z) = (z - \sigma_0)$ and take ξ_i as in (4.10). The ordering of the poles ensures that the basis functions $b_k(z)$ defined in Equation (4.12) vary only mildly on Σ, which avoids numerical under- or overflow. We summarise this procedure in Algorithm 4.4.

In Example 4.3 we show how the cyclic choice outperforms the implementation in the SLEPc module on a scalar example first appeared in [Güt+14].

Example 4.3. *Consider the scalar function* $g(z) = 0.2\sqrt{z} - 0.6\sin(2z)$. *The target set is* $\Omega = [10^{-2}, 4]$, *while* Σ *is formed by* 10^3 *logarithmically spaced points. Figure 4.2 shows four convergence plots corresponding to the approaches analysed earlier:*

- AAA (solid blue): *This plot shows the error* $\left\| g(z) - r^{(m)}(z) \right\|_\Sigma$ *of the interpolant obtained by AAA. An approximant of degree* 19 *achieves an error of* 10^{-14}, *although there are some spikes in the curve and a slight stagnation around* 10^{-13}.

- Leja–Bagby (dashed red with square markers): *This plot corresponds to the NLEIGS approach of section 4.2.1, which computes the Leja–Bagby points on* Σ *and on the singularity set* $\Xi =]-\infty, 0]$. *As explained in the above-mentioned section, the convergence rate is given in terms of the capacity of the condenser* $\operatorname{cap}(\Sigma, \Xi) \approx 0.569$

$$\limsup_{m\to\infty} \left\| g(z) - r^{(m)}(z) \right\|_{\Sigma} \leqslant \exp(-1/\operatorname{cap}(\Sigma, \Xi)).$$

- Leja–Bagby(10) + poles at infinity (dotted yellow with circles): *This is the approach used in [CR19]. In this example we fixed* 10 *iterations for the initial phase with the AAA algorithm, followed by a Leja–Bagby procedure with the d poles thus obtained, and then using poles at infinity for the remaining process. We see a drop in the error for degree* 10, *similar to the level of the AAA approximant, but then a really slow decay in the refinement phase when* $m > 10$. *This approach is equivalent to using a polynomial interpolation in the refinement phase. As explained in [Güt+14], we can expect a geometric convergence rate of only* $(\sqrt{k} - 1)/(\sqrt{\kappa} + 1) \approx 0.905$, *with* $\kappa = \max(\Sigma)/\min(\Sigma)$.

- 10-cyclic Leja–Bagby: *This is our proposed recommendation. Run an initial phase of AAA to get* $d = 10$ *poles, and then repeat them cyclically in Leja–Bagby order. In this way we combine the convenience of not having to specify the singularity set with the robust convergence of the Leja–Bagby approach.*

4.3.3 Surrogate AAA with cyclic Leja–Bagby refinement

The careful reader may have noted a subtle, but important flaw in both the methods explained in the previous section, whether the added poles lie at infinity or are repeated cyclically. We wrote that there are circumstances when surrogate AAA does not return a sufficiently precise approximation $R^{(d)}(z)$, therefore we performed a refinement based on the NLEIGS algorithm with the data retrieved from the first phase. However, we are discarding the approximant $R^{(d)}(z)$ even when it already

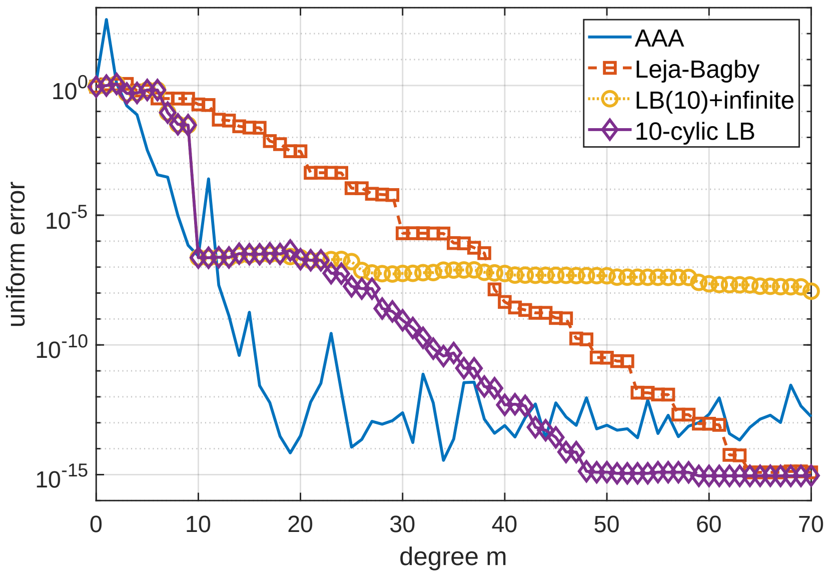


Figure 4.2: Demonstration of four different choices for the rational approximation of a scalar function, including our proposed d-cyclic Leja–Bagby procedure ($d = 10$).

satisfies (4.2) or is really close to. Hence, instead of constructing $R^{(m)}(z)$ from scratch with NLEIGS and the poles retrieved from the surrogate step, it is better to simply add additional terms to $R^{(d)}(z)$ and obtain an approximant of the form

$$R^{(m)}(z) = \sum_{i=0}^{d} \frac{G(\sigma_i)w_i}{z-\sigma_i} \Bigg/ \sum_{i=0}^{d} \frac{w_i}{z-\sigma_i} + b_{d+1}(z)R_{d+1} + \cdots + b_m(z)R_m \tag{4.40}$$

We named this approach *surrogate AAA with cyclic Leja–Bagby refinement*: its advantage with respect to Algorithm 4.4 is that we do not waste time in the Leja–Bagby refinement if the initial approximation is already good enough. We write the steps of its implementation here below and in Algorithm 4.5.

1. Run the surrogate AAA algorithm to compute the $d+1$ interpolation points σ_i and weights w_i for $i = 0, \ldots, d$. This returns a rational approximant $R^{(d)}(z)$ of type (d, d), where the degree d is determined by the AAA stopping criterion (4.22).

2. Compute the poles thanks to the linear problem (4.23) and reorder them thanks to (4.10) as explained in section 4.3.2, getting $\xi_1, \ldots, \xi_d$.

Algorithm 4.5: SURROGATE AAA WITH CYCLIC LEJA–BAGBY REFINEMENT.

```
Input: $G, \Sigma, m_{\max}, \varepsilon$
Output: $R^{(m)}(z)$.
1  Draw random vectors $u, v \in \mathbb{C}^n$
2  Build surrogate function $g(z) = u^* G(z) v$
3  $r^{(0)}(z) \leftarrow 0$, $n^{(0)}(z) \leftarrow 0$, $d^{(0)}(z) \leftarrow 1$
4  $\Sigma^{(0)} \leftarrow \Sigma$, $d \leftarrow 1$
5  while $d \leqslant m_{max}$ do
6      Compute $\sigma_d = \arg\max_{z \in \Sigma^{(d-1)}} \left| g(z) d^{(d-1)}(z) - n^{(d-1)}(z) \right|$
7      $\Sigma^{(d)} \leftarrow \Sigma^{(d-1)} \setminus \{\sigma_d\}$
8      Compute the SVD of matrix $A^{(d)}$ (4.21)
9      Retrieve the vector of weights $w$
10     Build $r^{(d)}(z)$
11     $d \leftarrow d + 1$
12     if $\left\| g - r^{(d-1)} \right\|_{\Sigma} < \varepsilon \|g\|_{\Sigma}$ or $d = m_{max} + 1$ then
13         continue
14 Compute the poles $\tilde{\Xi}$ from (4.23)
15 $\Xi \leftarrow$ sort $\tilde{\Xi}$ with (4.10) and compute $\beta_d$
16 $m \leftarrow d$
17 while $\|R_m\|_F > \varepsilon \max_{0 \leqslant k \leqslant m} \|G(\sigma_k)\|_F / 3$, and $m < m_{max}$ do
18     $\sigma_{m+1} \leftarrow \arg\max_{z \in \Sigma} |s_m(z)|$
19     $\xi_{m+1} \leftarrow \xi_{1+(k-1 \mod d)}$
20     $b_{m+1}(z) \leftarrow b_m(z)(z - \sigma_m)/(1 - z/\xi_{m+1})$
21     $\beta_{m+1}(z) \leftarrow \max_{z \in \Sigma} |b_m(z)|$
22     $b_{m+1}(z) \leftarrow b_{m+1}(z)/\beta_{m+1}$
23     Build $R_{m+1}$ with (4.13)
24     $m \leftarrow m + 1$
25 return $R^{(m)}(z)$
```

3. Apply the d-cyclic Leja–Bagby algorithm: for each new pair of node and pole $(\sigma_{d+i}, \xi_{d+i})$, where σ_{d+i} is computed as in (4.10) and ξ_{d+i} as in (4.39) build

$$R_{d+i} = \frac{G(\sigma_{d+i}) - R^{(d+i-1)}(\sigma_{d+i})}{b_{d+i}(\sigma_{d+i})}, \quad b_{d+i}(z) = \prod_{j=1}^{d+i} \frac{z - \sigma_{j-1}}{\beta_j (1 - z/\xi_j)}, \tag{4.41}$$

with β_j chosen such that $\|b_{d+i}\|_{\Sigma} = \max_{z \in \Sigma} |b_{d+i}(z)| = 1$.

If we are interested in the eigenvalues of $R^{(m)}(z)$, then its expression in (4.40) is not useful until we find a suitable linearization. Elsworth and Güttel showed that you can convert a rational approximant from the barycentric coordinates to the Newton basis [EG19]. Hence, a possibility is converting the first part of (4.40) and then use the linearization proposed in [Güt+14]. Here we suggest a mixed-form linearization, which avoids the conversion. First, rewrite $R^{(m)}(z)$ as

$$R^{(m)}(z) = \sum_{i=0}^{d} w_i G(z_i) b_i(z) + b_{d+1}(z) R_{d+1} + \cdots + b_m(z) R_m,$$

with $b_{d+i}(z)$, $i = 1, \dots m-d$, as in (4.13) and

$$b_i(z) = \frac{1}{z-\sigma_i} \Bigg/ \sum_{i=0}^{d} \frac{w_i}{z-\sigma_i}, \qquad i = 1, \dots, d.$$

It follows that

$$b_d(z) = \frac{1}{z-\sigma_d} \Bigg/ \sum_{i=0}^{d} \frac{w_i}{z-\sigma_i} = \prod_{j=0}^{d-1} (z-\sigma_j) \Bigg/ \sum_{i=0}^{d} w_i \prod_{j \neq i} (z-\sigma_j) \,.$$

The right-hand side has the same denominator of $R^{(d)}(z)$ in (4.33), up to a scalar multiplier. In addition, the numerator is the same one of the Newton basis function obtained after d iterations of Newton interpolation, as in (4.12). Hence $b_d(z)$ is a scalar multiple of the basis function obtained when using the Newton interpolation from the beginning. Therefore there is a recursion for all the basis functions $b_0(z), \dots, b_m(z)$, where the first d correspond to the barycentric basis, while the last $m-d$ to the Newton one:

$$\begin{aligned}
b_0(z) &= \frac{1}{z-\sigma_0} \Bigg/ \sum_{i=0}^{d} \frac{w_i}{z-\sigma_i}\,, \\
(z-\sigma_{i+1}) b_{i+1}(z) &= (z-\sigma_i) b_i(z), \quad i = 0, \dots, d-1, \\
\beta_{d+i+1}(1 - z/\xi_{d+i+1}) b_{d+i+1}(z) &= (z-\sigma_{d+i}) b_{d+i}(z), \quad i = 0, \dots, m-d-1.
\end{aligned}$$

We summarise the result in the following theorem.

Theorem 4.2. *Given the rational matrix-valued function $R^{(m)}(z)$ in (4.40), the rational eigenvalue problem $R^{(m)}(\lambda)v = 0$ is equivalent to $Ax = \lambda Bx$, where*

$$A = \begin{bmatrix} w_0F(\sigma_0) & w_1F(\sigma_1) & \cdots & w_dF(\sigma_d) & R_{d+1} & \cdots & R_{m-2} & R_{m-1} - \frac{\sigma_{m-1}}{\beta_m}R_m \\ \sigma_0 I & -\sigma_1 I & & & & & & \\ & \ddots & \ddots & & & & & \\ & & \sigma_{d-1}I & -\sigma_d I & & & & \\ & & & \frac{\sigma_d}{\beta_{d+1}}I & I & & & \\ & & & & \ddots & \ddots & & \\ & & & & & \ddots & \ddots & \\ & & & & & & \frac{\sigma_{m-2}}{\beta_{m-1}}I & I \end{bmatrix}$$

and

$$B = \begin{bmatrix} \frac{w_0F(\sigma_0)}{\xi_m} & \frac{w_1F(\sigma_1)}{\xi_m} & \cdots & \frac{w_dF(\sigma_d)}{\xi_m} & \frac{R_{d+1}}{\xi_m} & \cdots & \frac{R_{m-2}}{\xi_m} & \frac{R_{m-1}}{\xi_m} - \frac{R_m}{\beta_m} \\ I & -I & & & & & & \\ & \ddots & \ddots & & & & & \\ & & I & -I & & & & \\ & & & \frac{I}{\beta_{d+1}} & \frac{I}{\xi_{d+1}} & & & \\ & & & & \ddots & \ddots & & \\ & & & & & \ddots & \ddots & \\ & & & & & & \frac{I}{\beta_{m-1}} & \frac{I}{\xi_{m-1}} \end{bmatrix},$$

while $x = b(\lambda) \otimes v$ with $b(\lambda) = [b_0(\lambda)\, b_1(\lambda)\, \ldots, b_{m-1}(\lambda)]^T$.

4.4 ROBUSTNESS OF THE RATIONAL APPROXIMANTS

In this section we test how robust and stable are the algorithms described in sections 4.2 and 4.3. In order to have a broader range of problems, we also use matrix-valued functions that are *not* holomorphic in the target set and we see how the algorithms perform even outside their original scope. Hence, to underline this aspect, in this section we use the notation $F(z)$. We focus on the following algorithms:

1. If the matrix-valued function $F(z)$ is given in split form:

- the *set-valued AAA* algorithm of section 4.2.3 [Lie+18];
- the *weighted AAA* algorithm, which is the modification of the set-valued AAA with the new stopping criterion (4.27).

2. If the matrix-valued function $F(z)$ is given in black box form:

- the *surrogate AAA* of section 4.3.2, as proposed by Elsworth and Güttel [EG19];
- the *surrogate AAA with exact search on* $\partial\Sigma$ of section 4.3.1;
- the *NLEIGS with poles from surrogate AAA* described in [CR19] of section 4.3.2, where we repeat the poles cyclically as in (4.39) and the stopping criterion is (4.15);
- the *surrogate AAA with cyclic Leja–Bagby refinement* algorithm of section 4.3.3.

We are not including the Cauchy approximation of section 4.2.4 because we did not find a way to automatically choose an optimal contour for the Cauchy formula and because preliminary experiments showed that the degree of the approximant $R^{(m)}(z)$ returned by this method is much larger than the AAA one, as seen in Example 4.2.

In order to benchmark these algorithms we used the nonpolynomial problems from NLEVP 4.1, and we listed them in Table 4.3. We selected them to represent a variety of matrix-valued functions with different sizes and properties.

Experiment 1

In the first set of experiments we discretise the target sets Ω (either open disks or half open disks) as follows. We generate 300 uniformly distributed random points in Ω plus another 100 uniformly spaced points on $\partial\Omega$, for a total of 400 points for the set Σ. Given a tolerance ε and a problem in Table 4.3, we test if an algorithm fails to construct an approximant $R^{(m)}$ with accuracy $\|G - R^{(m)}\|_{\Sigma}/\|G\|_{\Sigma}$ below a given tolerance ε or if it does not converge within the maximum number of steps, which we set to 60. We also compare the degrees of the approximants. We report the results in Table 4.4 for the values $\varepsilon = 10^{-7}$, Tables 4.5 and 4.6 for $\varepsilon = 10^{-10}$, and Table 4.7 for $\varepsilon = 10^{-13}$.

Table 4.3: List of benchmark examples from the NLEVP collection, their type and size, the target set Ω (disc or half disc), and the number of eigenvalues in Ω. For the `canyon_particle` problem, $\gamma = -9 \times 10^{-2} + 10^{-6}\mathrm{i}$. The `fiber` and `sandwich_beam` problems are holomorphic on their respective target set if we remove the negative real numbers. Similarly, `schrodinger_abc` is holomorphic on $\Omega \backslash [-15, -10[$.

Name	type	size	center	radius	half disc	#evs	holomorphic
`bent_beam`	nonlinear	6	60	30	yes	2	yes
`buckling_plate`	nonlinear	3	11	9	no	12	no
`canyon_particle`	square root	55	γ	0.1	yes	15	yes
`clamped_beam_1d`	exponential	100	0	10	no	101	yes
`distributed_delay1`	nonlinear	3	0	2	no	2	yes
`fiber`	nonlinear	2400	0	0.002	yes	1	no
`gun`	square root	9956	62500	50000	yes	21	yes
`hadeler`	exponential	200	-30	11.5	no	14	yes
`loaded_string`	rational	100	362	358	no	9	yes
`nep1`	nonlinear	2	0	3	no	6	yes
`nep2`	nonlinear	3	0	2	no	4	yes
`nep3`	nonlinear	10	5i	2	no	14	yes
`neuron_dde`	exponential	2	0	15	no	11	yes
`pdde_symmetric`	exponential	81	0	2	no	59	yes
`photonic_crystal`	nonlinear	288	11	9	no	28	yes
`pillbox_small`	square root	20	0.08	0.05	yes	1	yes
`railtrack2_rep`	rational	1410	3	2	no	53	yes
`railtrack_rep`	rational	1005	-3	2	no	2	yes
`sandwich_beam`	nonlinear	168	7000	6900	no	7	yes
`schrodinger_abc`	nonlinear	10	-10	5	no	6	no
`square_root`	square root	20	10+50i	50	no	3	yes
`time_delay`	exponential	3	0	15	no	8	yes
`time_delay2`	exponential	2	0	15	no	11	yes
`time_delay3`	exponential	10	2	3	no	38	yes

We report the results for the rational problems `loaded_string`, `railtrack2_rep`, and `railtrack_rep` only in Table 4.4 since for these problems and any tolerance $\varepsilon \leqslant 10^{-5}$, all the algorithms return (rightly) a degree 2 rational approximant with a relative error of about 10^{-15}. The tables show that for our test problems and tolerances $\varepsilon \in \{10^{-7}, 10^{-10}, 10^{-13}\}$:

1. The set-valued and weighted AAA algorithms always return an approximant $R^{(m)}(z)$ with relative error below the required accuracy. For problems that are holomorphic on the target sets, surrogate AAA with cyclic Leja–Bagby refinement and NLEIGS with poles from surrogate AAA also return an approximant $R^{(m)}(z)$ with relative error below the required accuracy.

2. The set-valued and weighted AAA algorithms typically return the approximants $R^{(m)}(z)$ of lowest degrees. The degrees of the set-valued AAA and weighted AAA approximants are more or less the same: they are usually either equal or they differ by one. There are exceptions though such as with the `sandwich_beam` and `time_delay3` problems, for which weighted AAA returns a lower degree approximant. These two problems have the particularity that, when viewed in split form, the norms of their coefficient matrices have large variations. The latter is exploited by the weighted AAA algorithm, but ignored by the set-valued AAA algorithm.

3. The surrogate AAA approach often fails to return a rational approximant with relative accuracy below ε. There is no surprise here since there is no guarantee of any accuracy with the stopping criterion used by this algorithm.

4. As expected by our analysis, surrogate AAA with exact search either returns a rational approximant with relative error below the tolerance or fails to converge. There is an exception though for the `fiber` problem and tolerance $\varepsilon = 10^{-7}$ (see Table 4.4), where the constructed rational approximant $R^{(m)}$ is such that $\|F - R^{(m)}\|_\Sigma / \|F\|_\Sigma = 1.6 \times 10^{-7} > \varepsilon$, and hence marked as failed in the table. The reason why the relative error is slightly above the tolerance is that the exact search is done on $\partial\Sigma = \Sigma \cap \partial\Omega$ and since $F(z)$ is not holomorphic on

Ω, $\|\cdot\|_\Sigma \geqslant \|\cdot\|_{\partial\Sigma}$ so the stopping criterion (4.35) does not guarantee that (4.2) holds. However, if we run the surrogate AAA with exact search on Σ in place of $\partial\Sigma$, then we get a rational approximant $R^{(m)}(z)$ with relative error 4×10^{-8}, i.e., below the tolerance $\varepsilon = 10^{-7}$.

Furthermore, Tables 4.4, 4.6 and 4.7 show that for a few problems, the algorithm reaches the maximum number of steps, i.e., 60 (indicated by red stars) while returning approximants of degree less than 60. This is due to the removal of the Froissart doublets (see Remark 4.3).

5. NLEIGS with poles from surrogate AAA repeated in a cyclic way has a behaviour similar to that of surrogate AAA with cyclic Leja–Bagby refinement: they typically return rational approximants of the same degree, and they only fail for $\varepsilon = 10^{-13}$ for the `buckling_plate`, since it is not holomorphic on the target set and the theoretical results do not hold.

Experiment 2

In this second experiment we visualise where the algorithms place the interpolation nodes σ_i and the poles ξ_i in and around the target set Ω. We are doing this on a subset of the problems in Table 4.3 and we only display the nodes and poles for the weighted AAA and surrogate AAA with cyclic Leja–Bagby refinement (Figure 4.3). For the latter, we distinguish the nodes chosen by the surrogate AAA phase from the nodes chosen by the refinement phase and we leave out the poles that are too far from Ω. The discretization of Ω consists of 300 points randomly generated inside Ω plus another 50 points uniformly distributed on the contour $\partial\Omega$. The maximum number of steps is set to 60 and the tolerance to $\varepsilon = 10^{-10}$.

The results are already anticipated by the theory. For the holomorphic problems, the nodes lie on the contour $\partial\Omega$, while the poles form a pattern outside Ω. For instance, they are aligned towards the branch points for the `gun` problem, which contain two square roots. For the `nep1` problem, both algorithms use the same interpolation

Table 4.4: Degree of $R^{(m)}(z)$ for $\varepsilon = 10^{-7}$ and 24 problems. The lowest degrees are highlighted in bold/blue, including any within one of the lowest. We excluded those corresponding to a failed required accuracy and we provided them within square brackets. A '$\star$' indicates that the algorithm reached the maximum number of steps.

Problem	set-valued AAA	weighted AAA	surrogate AAA	surrogate+ exact search	NLEIGS AAA poles	surrogate+ LB refine
`bent_beam`	**7**	**7**	[4]	**8**	11	10
`buckling_plate`	**23**	**23**	[21]	26	42	42
`canyon_particle`	**13**	**12**	[7]	17	20	20
`clamped_beam_1d`	**11**	**11**	[10]	16	23	24
`distributed_delay1`	**6**	**6**	[6]	9	11	10
`fiber`	13	**10**	[6]	[16]	22	21
`gun`	**9**	**9**	[5]	[60]*	15	15
`hadeler`	**2**	4	**2**	6	15	5
`loaded_string`	**2**	**2**	**2**	**2**	**2**	**2**
`nep1`	**21**	**20**	**20**	**20**	**20**	**20**
`nep2`	**13**	**13**	[9]	[13]*	20	19
`nep3`	**8**	**8**	[7]	10	16	16
`neuron_dde`	**13**	**13**	[12]	**13**	**14**	**14**
`pdde_symmetric`	**8**	**8**	[7]	11	16	15
`photonic_crystal`	**5**	**5**	**4**	**5**	9	9
`pillbox_small`	**7**	**6**	[4]	9	11	11
`railtrack2_rep`	**2**	**2**	**2**	**2**	**2**	**2**
`railtrack_rep`	**2**	**2**	**2**	**2**	**2**	**2**
`sandwich_beam`	10	6	**3**	8	11	10
`schrodinger_abc`	**11**	**11**	[10]	16	20	20
`square_root`	**9**	**10**	**9**	**10**	21	14
`time_delay`	**12**	**13**	**12**	**12**	**13**	**12**
`time_delay2`	**12**	**13**	**12**	**12**	**13**	**12**
`time_delay3`	16	**13**	**12**	**12**	**13**	**12**
# of fails	0	0	13	3	0	0
# of lowest degree	21	22	11	11	8	8

Table 4.5: Accuracy $\left\|F - R^{(m)}\right\|_\Sigma / \|F\|_\Sigma$ for $\varepsilon = 10^{-10}$ and 21 problems. Any relative error above ε is highlighted in red and considered as a fail. A '★' indicates that the algorithm reached the maximum number of steps, i.e., 60.

Problem	set-valued AAA	weighted AAA	surrogate AAA	surrogate+ exact search	NLEIGS AAA poles	surrogate+ LB refine
`bent_beam`	4e-11	7e-12	2e-06	8e-07★	1e-12	2e-11
`buckling_plate`	3e-11	2e-11	1e-07	2e-06★	3e-11	5e-11
`canyon_particle`	4e-12	1e-11	2e-06	7e-08★	2e-11	1e-11
`clamped_beam_1d`	1e-11	1e-11	1e-06	4e-09★	3e-13	5e-13
`distributed_delay1`	1e-12	5e-13	3e-06	1e-08★	2e-12	2e-12
`fiber`	4e-13	3e-11	2e-06	1e-06★	1e-11	2e-11
`gun`	4e-13	3e-13	6e-07	2e-07★	1e-12	1e-12
`hadeler`	2e-11	5e-12	4e-10	2e-11	2e-12	4e-12
`nep1`	1e-11	9e-12	9e-12	9e-12	3e-11	9e-12
`nep2`	3e-12	3e-12	2e-05	5e-06★	1e-11	4e-12
`nep3`	5e-11	4e-12	2e-07	2e-09★	9e-12	1e-11
`neuron_dde`	6e-11	3e-11	8e-09	7e-11	8e-12	2e-11
`pdde_symmetric`	4e-11	3e-13	2e-07	2e-09★	2e-12	4e-12
`photonic_crystal`	6e-16	7e-16	1e-15	1e-15	2e-15	1e-15
`pillbox_small`	3e-13	8e-12	3e-07	1e-09★	1e-11	7e-12
`sandwich_beam`	5e-16	6e-12	6.8e-9	5.2e-09★	3.3e-12	9.6e-13
`schrodinger_abc`	2e-11	5e-12	1e-07	2e-08★	5e-12	7e-12
`square_root`	6e-12	4e-12	2e-12	2e-12	3e-12	2e-12
`time_delay`	3e-11	7e-12	7e-11	1e-11	8e-12	1e-10
`time_delay2`	3e-11	2e-12	2e-09	4e-11	8e-12	9e-11
`time_delay3`	6e-12	4e-12	4e-10	4e-11	7e-12	2e-11
# of fails	0	0	17	13	0	0

Table 4.6: Degree of $R^{(m)}(z)$ for $\varepsilon = 10^{-10}$ and 21 problems. The lowest degrees are highlighted in bold/blue including any within one of the lowest and excluding those corresponding to a failed required accuracy that are provided within square brackets). A '★' indicates that the algorithm reached the maximum number of steps, i.e., 60.

Problem	set-valued AAA	weighted AAA	surrogate AAA	surrogate+ exact search	NLEIGS AAA poles	surrogate+ LB refine
`bent_beam`	**9**	**9**	[6]	[23]★	13	12
`buckling_plate`	**26**	**27**	[24]	[35]★	47	48
`canyon_particle`	**18**	**17**	[11]	[17]★	29	30
`clamped_beam_1d`	**13**	**13**	[12]	[16]★	29	29
`distributed_delay1`	**8**	**8**	[7]	[9]★	15	15
`fiber`	18	**15**	[11]	[19]★	30	30
`gun`	**12**	**12**	[8]	[60]★	21	21
`hadeler`	**7**	**8**	[6]	23	21	19
`nep1`	**25**	**24**	**24**	**24**	**24**	**24**
`nep2`	**16**	**16**	[11]	[14]★	22	22
`nep3`	**10**	**10**	[9]	[11]★	17	17
`neuron_dde`	**16**	**15**	[14]	18	31	31
`pdde_symmetric`	**9**	**10**	[9]	[11]★	18	18
`photonic_crystal`	**7**	**6**	**6**	**6**	**6**	**6**
`pillbox_small`	**10**	**9**	[7]	[11]★	16	16
`sandwich_beam`	14	**8**	[7]	[60]★	18	20
`schrodinger_abc`	**13**	**13**	[12]	[60]★	21	21
`square_root`	**13**	**13**	**13**	**13**	**13**	**13**
`time_delay`	**15**	**15**	**14**	**15**	31	20
`time_delay2`	**15**	**16**	[14]	17	31	24
`time_delay3`	20	**16**	[14]	**16**	31	31
# of fails	0	0	17	13	0	0
# of lowest degree	18	21	4	5	3	3

Table 4.7: Degree of $R^{(m)}(z)$ for $\varepsilon = 10^{-13}$ and 21 problems. The lowest degrees are highlighted in bold/blue including any within one of the lowest and excluding those corresponding to a failed required accuracy that are provided within square brackets). A '★' indicates that the algorithm reached the maximum number of steps, i.e., 60.

Problem	set-valued AAA	weighted AAA	surrogate AAA	surrogate+ exact search	NLEIGS AAA poles	surrogate+ LB refine
`bent_beam`	**12**	**11**	[7]	[23]★	16	16
`buckling_plate`	**30**	**30**	[26]	[35]★	[60]★	[60]★
`canyon_particle`	**23**	**22**	[16]	[18]★	38	39
`clamped_beam_1d`	**15**	**15**	[14]	[16]★	32	32
`distributed_delay1`	**9**	**9**	[8]	[9]★	16	16
`fiber`	22	**20**	[16]	[19]★	42	38
`gun`	**15**	**15**	[11]	[60]★	27	27
`hadeler`	**10**	**11**	[9]	[59]★	23	23
`nep1`	**29**	**28**	**28**	**28**	**28**	**28**
`nep2`	**18**	**19**	[12]	[14]★	24	24
`nep3`	**12**	**12**	[10]	[11]★	22	22
`neuron_dde`	**19**	**18**	[17]	[60]★	30	32
`pdde_symmetric`	**11**	**11**	[11]	[11]★	21	20
`photonic_crystal`	**7**	**6**	**6**	**6**	**6**	**6**
`pillbox_small`	**13**	**12**	[9]	[11]★	20	22
`sandwich_beam`	17	**12**	[11]	[60]★	22	22
`schrodinger_abc`	**15**	**15**	[13]	[60]★	26	26
`square_root`	**16**	**16**	**15**	**16**	33	**15**
`time_delay`	**18**	**18**	[17]	23	24	22
`time_delay2`	**17**	**18**	[17]	[60]★	28	29
`time_delay3`	23	**19**	[17]	[60]★	27	24
# of fails	0	0	18	17	1	1
# of lowest degree	18	21	3	3	2	3

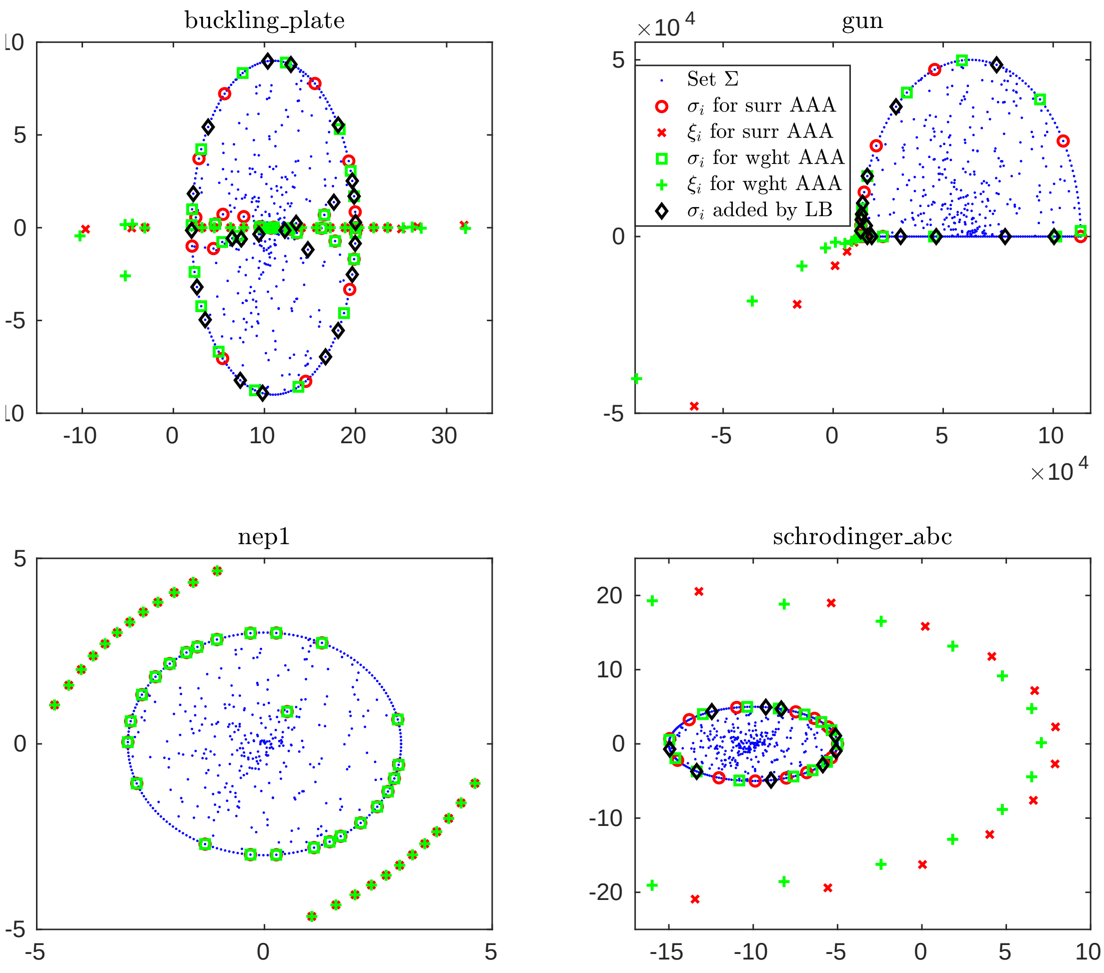


Figure 4.3: The set Σ, the interpolation nodes σ_i and the poles ξ_i nearest to Σ for a subset of the problems in Table 4.3.

nodes and poles. This happens because there is only one function to approximate, as one can see from the its split form in Example 4.1. Interestingly, one of the nodes lies inside Ω despite the problem being holomorphic. Finally, the poles and the nodes of `buckling_plate` do not follow the same pattern, because this problem is not holomorphic in the chosen region.

4.5 COMPARISON WITH THE CONTOUR SOLVER ALGORITHM

In Chapter 2 we generalised contour algorithms to solve meromorphic problems on a target set Ω. In Chapter 3 we showed how to choose the many parameters that this kind of algorithms needs and how to automatically refine the eigenpairs in the case the core of the algorithm does not return satisfactory results. Finally, in Chapter 4 we proposed robust ways to approximate nonlinear functions with rational ones with the idea of solving the latter problems through a linearization. The goal of this last set of numerical experiments is wrapping up everything we discussed and compare the results of these two methods.

We are aware the comparison is not the fairest possible. As we mentioned more than once, algorithms based on rational approximations thrive with large, sparse problems, while for contour algorithms we focused on smaller, but dense matrices. In addition, one has to analyse the data with a grain of salt, even more the one concerning the timings. For instance, the contour solver does not exploit parallelization anywhere, while the rational approximation algorithms use `eig` or `eigs` as final step. Nevertheless, we believe this set of experiments give us good insights on the strengths and weaknesses of the two approaches.

The specific algorithms we compare are:

- the weighted AAA (Section 4.2.3);
- the surrogate AAA with cyclic Leja–Bagby refinement (Section 4.3.3);
- the Hankel interpretation of the contour solver (Section 2.4.1);
- the Loewner interpretation of the contour solver (Section 2.4.2);
- the Hankel interpretation of the contour solver with the automatic refinement. More specifically, if the backward error of some eigenpairs is larger than the given threshold, these eigenpairs are then refined with either the Newton itera-

tion or the subspace update. These refinements and the choice of one over the other are explained in more details in Section 3.5.

First, note that we only compare the weighted AAA and the surrogate AAA with cyclic Leja–Bagby refinement because they have already performed the best among their competitors, i.e., rational approximation algorithms. Furthermore, we point out that weighted AAA is the only one that exploits the split form of the matrix-valued functions. Let us now describe the parameters used. In the rational approximation algorithms, we drew 300 uniformly random points inside Ω and 50 uniformly distributed on $\partial\Omega$. The error of the approximation was set to 10^{-7} and we computed the eigenvalues thanks to built-in MATLAB functions `eig` or `eigs`. The problems where we have called the latter are highlighted in Table 4.8. In these cases, we asked to compute all the eigenvalues and the shift σ was given by the center of Ω. We then removed all the eigenvalues that were outside Ω and all the eigenvalues whose backward error was larger than 10^{-3}. Concerning the contour solver algorithms, the number of trapezoidal points N for disk of radius ρ is given by the formula

$$\max\{\lceil 2^{(5+\log_{10}(\rho))}\rceil, 4\}$$

We found this heuristic from preliminary experiments and it usually works sufficiently well, even though the theory warns us against it (see Section 3.4). For the Loewner interpretation, for a target set $\Omega = \mathcal{D}(\gamma, \rho)$, the left and right interpolation points σ_i lie intervowen on a semicircle with center γ and radius ρ_1 that satisfies (3.19), i.e.,

$$|\rho_1 - \gamma| < \rho\mu^{-1/N},$$

where $\mu \approx 10^{-16}$ is the machine precision. All the other parameters are chosen at runtime by the algorithm. Finally, the careful reader would point out that the backward error of the eigenpairs returned by the algorithms is not exactly the same. In fact, while the numerator in $\eta_F(\lambda, v)$ is always fixed, i.e., $\|F(\lambda)v\|_2$, the denominator changes slightly, since it is an approximation of $\|F\|_\Omega$. Nevertheless, in order to be as precise as possible, in this section we compute the backward error as explained in [GT17, Section 2]:

$$\eta_F(\lambda, v) = \frac{\|F(\lambda)v\|_2}{\|v\|_2 \sum_{j=1}^{s} \|A_j\|_F |f_j(\lambda)|}.$$

Table 4.8 does not include the problems `fiber`, `gun` and `sandwich_beam`, because the rational approximation algorithms were not able to return the eigenpairs in a reasonable amount of time.

The results of Table 4.8 are also available as a plot in Figure 4.4. By only comparing the number of returned eigenvalues, *weighted AAA* is the best performing algorithm, with only a single problem (`time_delay3`) where it fails to return the correct number of eigenvalues. On the opposite side we have the surrogate AAA and the contour Loewner, while in the middle lie the contour Hankel one and its refined version. The difference in the performances between weighted and surrogate stands in the larger degree of the approximation $R^{(m)}(z)$ that (usually) the surrogate algorithm requires. On the other hand, the Loewner interpretation usually performs worse than its original counterpart due to the location of the sampling poles. It is reasonable to assume that for specific situations one may find sampling points such that the returned eigenvalues have a smaller backward error. However, trying to automate this in a general case seems far too difficult for our current knowledge.

Looking at the problems where the contour Hankel algorithm (and its refinement) "fail" is far more interesting. First, we have `schrodinger_abc` and `pillbox_small`. This is expected, since these two problems contain square roots and thus cannot be meromorphic in a disk that contains the real line. Then there is `pillbox_small`: it has an eigenvalue just outside the contour, hence the algorithm computes a numerical rank equal to 29. This would not have happened with just some more quadrature points. Finally, we have `buckling_plate`, `nep1`, and `time_delay`. These are the most interesting ones. They are identified by the functions

$$F_1(z) = \begin{bmatrix} \frac{z(1-2z\cot 2z)}{\tan z - z} + 10 & \frac{z(2z-\sin 2z)}{\sin 2z(\tan z - z)} & 2 \\ \frac{z(2z-\sin 2z)}{\sin 2z(\tan z - z)} & \frac{z(1-2z\cot 2z)}{\tan z - z} + 4 & 2 \\ 2 & 2 & 8 \end{bmatrix}, \quad F_2(z) = \begin{bmatrix} e^{iz^2} & 1 \\ 1 & 1 \end{bmatrix},$$

$$F_3(z) = \begin{bmatrix} -z & 1 & 0 \\ 0 & -z & 1 \\ -(a_3 + b_3 e^{-z}) & -(a_2 + b_2 e^{-z}) & -(z + a_1 + b_1 e^{-z}) \end{bmatrix}.$$

As explained in Section 2.4.1 and 2.4.3 the contour solver algorithm has two "silent" hypotheses. For sake of simplicity, we remind them only in the holomorphic case. If the number of eigenvalues, say s, is less than the size of the problem, say n, then one assumes that the eigenvectors are independent (see (2.35)); otherwise, if $s > n$, then we know we can choose a number of moments m and a matrix P (usually the identity) such that $\operatorname{rank}(B_0) = s$ (see (2.40)). The silent hypothesis here is assuming that (2.40) is satisfied for $m > sn^{-1}$. This is not true for the three problems above. At runtime the algorithm correctly identifies $s_2 = 6$, $s_3 = 8$ for the holomorphic problems, and therefore chooses $m_2 = 4$, and $m_2 = 3$. For `buckling_plate`, the estimation of the number of eigenvalues (minus poles) is 1, but it keeps going because the matrix B_0 is not rank deficient and stops with $m_1 = 2$ and $P = I$. As expected, the final B_0 is rank deficient in all three cases, hence the algorithm does not update the parameters and proceeds with the eigenvalue computation. Its rank is indeed strictly less than s_i (in the holomorphic cases), and thus one may think that we could add further moments until it is equal to s_i. Unfortunately, this scenario is indistinguishable from the one where we overestimate the number of eigenvalues in Ω, hence this solution is not practical. The minimum values of m_i which return the correct number of eigenvalues are $m_1 = 7$, $m_2 = 6$, and $m_3 = 6$, respectively. In those cases, the backward error has order equal to the machine precision.

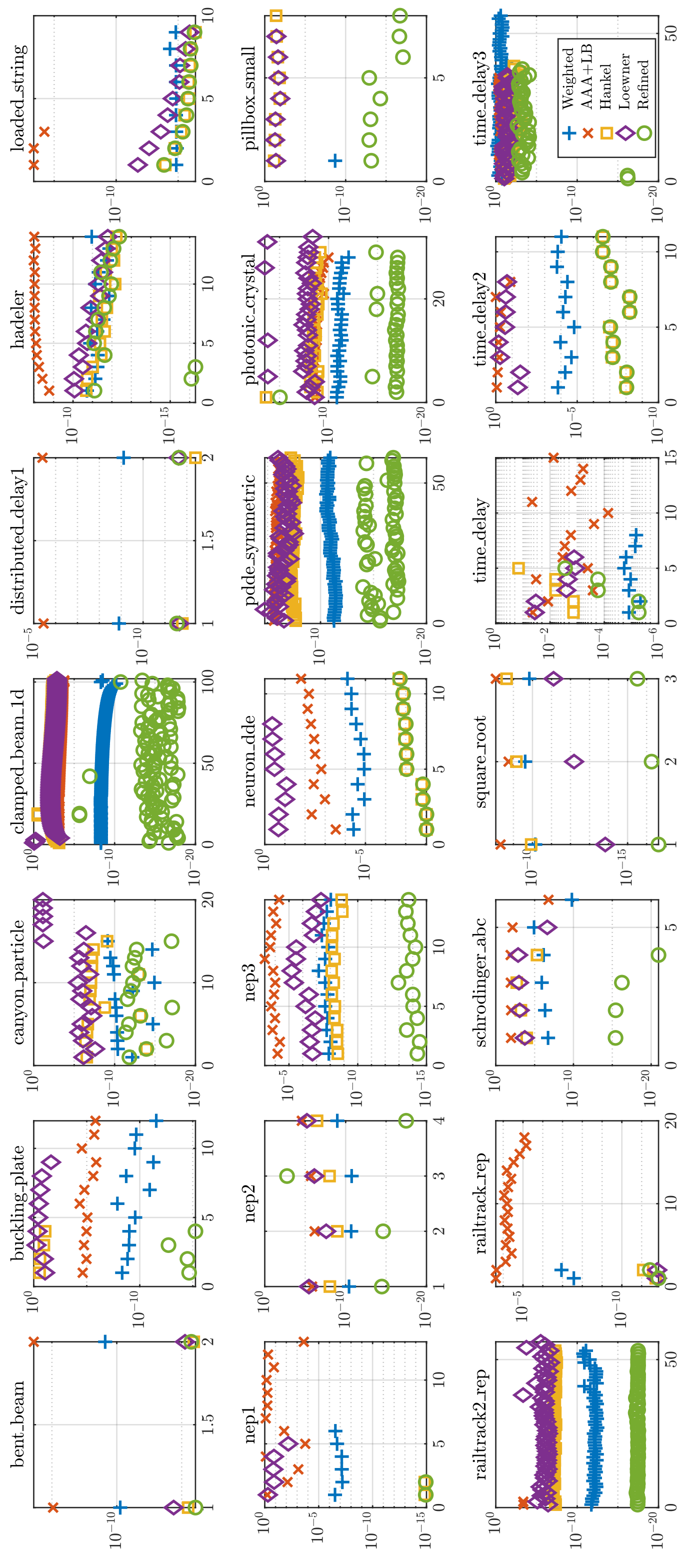


Figure 4.4: The backward error for all the algorithms and the problems in Table 4.8.

Table 4.8: Number of eigenvalues retrieved by the algorithms, with maximum and minimum backward error. In purple we highlighted the problems where the rational approximant algorithms used eigs as eigensolver. In red we highlighted the entries where the incorrect number of eigenvalues is returned. See also Figure 4.4.

Problem	weighted AAA			Surrogate AAA			Contour Solver			Contour Loewner			Contour Refined		
	# Evs	max	min	# Evs	max	min	# Evs	max	min	# Evs	max	min	# Evs	max	min
bent_beam (2)	2	2e-10	7e-10	2	3e-05	2e-04	2	3e-16	4e-16	2	2e-15	4e-15	2	7e-17	7e-16
buckling_plate (12)	12	5e-12	5e-08	12	1e-06	4e-05	4	1e-01	2e-01	9	2e-02	6e-01	4	7e-16	3e-13
canyon_particle (15)	15	1e-14	8e-09	0	Inf	Inf	15	1e-13	4e-06	20	3e-07	7e-01	15	8e-17	4e-11
clamped_beam_1d (101)	101	6e-12	7e-10	101	2e-05	5e-05	101	7e-05	3e-02	102	6e-05	4e-01	101	8e-20	3e-07
distributed_delay1 (2)	2	8e-10	1e-09	2	4e-06	4e-06	2	4e-13	2e-12	2	2e-12	3e-12	2	2e-12	3e-12
hadeler (14)	14	2e-11	4e-10	14	3e-08	2e-07	14	1e-11	4e-10	14	3e-11	2e-09	14	9e-16	1e-10
loaded_string (9)	9	1e-14	5e-14	3	6e-04	4e-03	9	4e-16	1e-13	9	1e-15	2e-11	9	5e-16	1e-13
nep1 (6)	6	2e-10	8e-10	13	9e-04	2e-03	2	4e-18	4e-18	5	2e-05	1e-03	2	3e-18	3e-18
nep2 (4)	4	2e-11	3e-09	4	8e-07	6e-05	4	1e-09	8e-07	4	3e-08	1e-05	4	8e-18	4e-03
nep3 (14)	14	1e-09	5e-09	14	5e-06	5e-05	14	1e-10	8e-10	14	5e-09	7e-07	14	3e-16	8e-15
neuron_dde (11)	11	2e-10	1e-09	11	8e-09	1e-07	11	2e-13	1e-12	8	2e-06	8e-06	11	2e-13	1e-12
pdde_symmetric (59)	59	2e-11	3e-10	59	5e-06	2e-05	59	9e-08	1e-06	59	3e-07	3e-04	59	5e-17	1e-13
photonic_crystal (28)	28	1e-11	2e-10	28	1e-09	7e-08	29	8e-09	3e-03	32	4e-08	5e-03	29	1e-16	1e-04
pillbox_small (1)	1	1e-08	1e-08	0	Inf	Inf	8	8e-02	4e-01	7	7e-02	3e-01	8	5e-17	7e-13
railtrack2_rep (53)	53	1e-11	7e-10	2	1e-02	2e-02	53	2e-06	5e-06	56	1e-05	2e-02	53	1e-16	3e-16
railtrack_rep (2)	2	8e-07	4e-06	18	2e-04	8e-03	2	6e-11	3e-10	2	5e-11	5e-11	2	4e-11	1e-10
schrodinger_abc (6)	6	3e-11	2e-09	6	1e-06	3e-06	4	2e-08	3e-05	5	4e-08	1e-06	4	7e-21	7e-20
square_root (3)	3	4e-10	2e-09	3	1e-08	5e-08	3	8e-10	1e-08	3	1e-13	4e-11	3	2e-16	2e-15
time_delay (8)	8	2e-11	5e-11	15	5e-09	6e-04	5	1e-08	2e-07	6	5e-09	4e-07	5	2e-11	5e-09
time_delay2 (11)	11	2e-09	9e-09	8	7e-06	2e-02	11	4e-13	7e-12	8	5e-06	5e-05	11	4e-13	7e-12
time_delay3 (38)	56	3e-11	5e-09	35	2e-11	3e-03	39	1e-13	1e-03	36	1e-12	1e-10	38	1e-23	6e-13

Table 4.9: The timings. In red we highlighted the entries larger than 10 seconds. See also Figure 4.5 for a visual representation.

Problem	weighted AAA	Surrogate AAA	Contour Solver	Contour Loewner	Contour Refined
`bent_beam (2)`	9.77e-02	6.90e-02	1.38e-01	1.19e-01	1.14e-01
`buckling_plate (12)`	3.93e-02	6.10e-02	6.03e-02	1.13e-01	5.82e-02
`canyon_particle (15)`	6.67e+00	3.00e+01	1.52e-01	3.37e-01	1.33e-01
`clamped_beam_1d (101)`	2.04e+01	1.52e+02	4.09e-01	1.09e+00	4.30e-01
`distributed_delay1 (2)`	3.52e-01	1.97e-01	1.85e-01	1.85e-01	1.88e-01
`hadeler (14)`	9.70e+00	8.21e+00	1.06e+00	1.67e+00	1.07e+00
`loaded_string (9)`	2.58e-01	1.14e-01	9.11e-01	1.08e+00	9.98e-01
`nep1 (6)`	3.13e-02	4.32e-02	3.86e-02	6.75e-02	3.64e-02
`nep2 (4)`	3.98e-02	3.45e-02	4.18e-02	6.14e-02	4.27e-02
`nep3 (14)`	4.23e-02	6.45e-02	7.28e-02	1.29e-01	5.59e-02
`neuron_dde (11)`	3.86e-02	4.53e-02	8.51e-02	1.44e-01	6.58e-02
`pdde_symmetric (59)`	3.41e+00	1.94e+01	6.93e-01	1.23e+00	7.16e-01
`photonic_crystal (28)`	5.70e+01	1.51e+02	5.86e+00	7.51e+00	6.42e+00
`pillbox_small (1)`	3.13e+01	8.27e+01	5.36e-02	7.23e-02	4.40e-02
`railtrack2_rep (53)`	1.18e+03	2.97e+02	7.14e+01	8.13e+01	6.57e+01
`railtrack_rep (2)`	3.71e+02	7.26e+01	1.52e+01	4.57e+01	4.05e+01
`schrodinger_abc (6)`	6.24e-02	1.11e-01	9.80e-02	1.17e-01	8.80e-02
`square_root (3)`	9.30e-02	1.21e-01	2.20e-01	3.25e-01	2.04e-01
`time_delay (8)`	1.80e-02	1.44e-02	5.56e-02	9.54e-02	5.61e-02
`time_delay2 (11)`	2.08e-02	1.72e-02	5.68e-02	1.22e-01	5.37e-02
`time_delay3 (38)`	5.43e-02	5.36e-02	8.05e-02	2.17e-01	6.62e-02

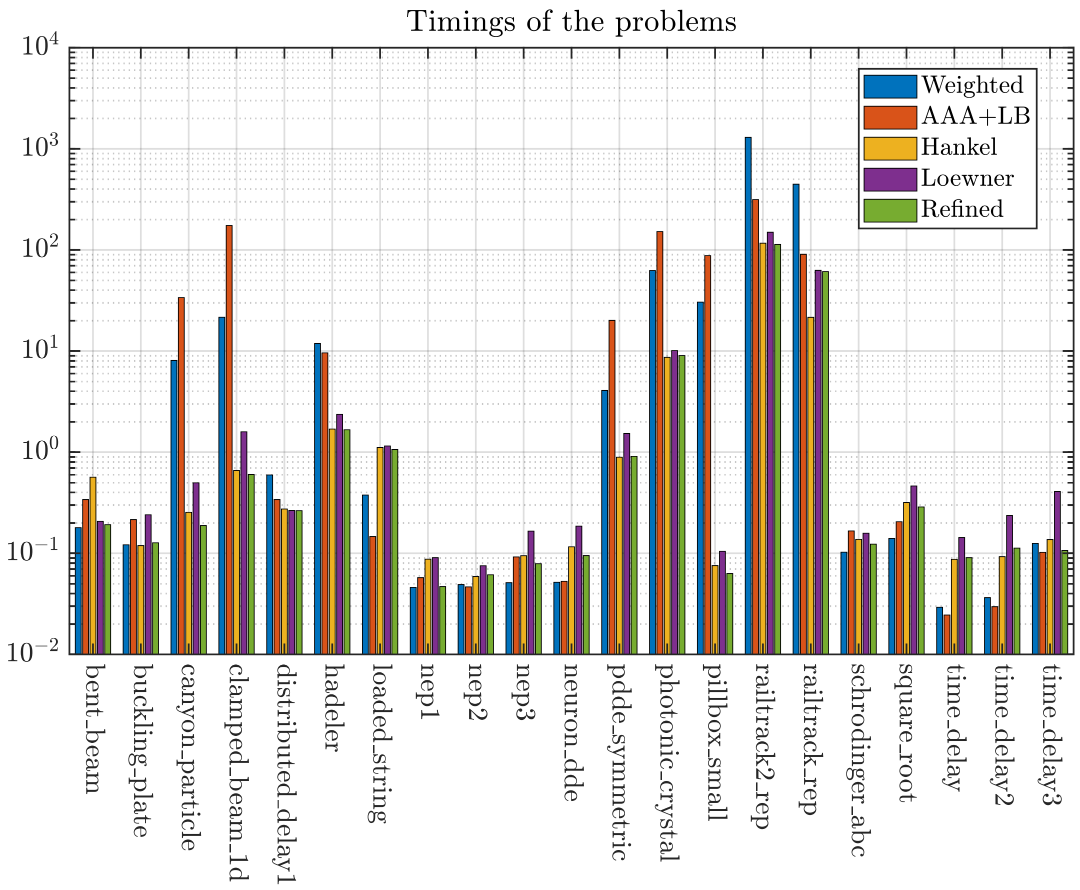


Figure 4.5: The timings in second for the algorithms to return the eigenpairs. We can see that the contour algorithms are generally faster. See also Table 4.9.

4.6 FINAL REMARKS

In this chapter we focused on rational approximation algorithms to solve nonlinear eigenvalue problems. In the first part, we formalised the relationship between the backward error of the eigenvalues of an approximant $R^{(m)}(z) \approx G(z)$ and the real eigenvalues of $G(z)$, and thus proved that using a rational approximant to compute the eigenvalues of a nonlinear function is a viable strategy.

In the second part, we described several state of the art algorithms and proposed numerous improvements. Among them, two algorithms were the clear winners. For matrix-valued functions in split form, the weighted AAA algorithm is a robust procedure to approximate holomorphic functions $G(z)$: it is scaling independent and returns an approximant with a user-chosen accuracy on the discretised target set Σ. We achieve this goal through a stopping criterion that includes weights relative to the importance of each scalar function in the split form.

For holomorphic black box functions, the two-phase " surrogate AAA algorithm with cyclic Leja–Bagby refinement" performs best. While it is more computationally expensive than the weighted AAA algorithm, it only requires the ability to evaluate the matrix-valued function at the point in the target set. Once more, it is scaling independent and returns a rational approximant with a user chosen accuracy on the discretized target set Σ, as long as Σ contains enough points. It combines the strength of surrogate AAA to identify good pole parameters in the first phase with the robustness of the Leja–Bagby approach in the second phase.

Finally, we compare these algorithms with the contour solvers of Chapters 2 and 3 with the nonlinear problems of the NLEVP collection. This extensive set of examples show the positive and negative sides of each approach. On one hand, the current implementation of the contour solvers is faster, requires fewer parameters to set and does not depend on a good approximation of the function. On the other, there exist problems where the necessary hypotheses on the independence of the eigenvectors are not satisfied, and thus an automatic choice of the settings is not able to return the eigenvalues up to the desired precision. Therefore, the end user still needs to understand the theory behind the algorithm to avoid unpleasant and unexpected results.

This chapter marks the end of our contributions to nonlinear eigenvalue problems. There are two directions towards which we should push our future works. For the rational approximations algorithms we should develop specific eigensolvers that exploit the structure of the linearizations, instead of simply relying on `eig` or `eigs`. As explained in the introduction to this chapter, this is the last part of every rational approximation algorithm, which we overlooked in this thesis. Another possibility is exploring further generalisations of the surrogate AAA algorithm: instead of using it as a first step in a two-phase algorithm, an interesting question is asking what would happen if we consider "higher dimensional" surrogate functions. More specifically, in this thesis we took two vectors $u, v \in \mathbb{C}^n$ and built the surrogate function $u^*G(z)v$. However, nothing should stop us in taking $u, v \in \mathbb{C}^{n\times k}$, with $k \ll n$, and approximating a $k \times k$ matrix-valued function. In this case it is not immediately clear how to go from the weights and sample points of the surrogate function to the ones of $G(z)$, so this approach requires indeed further studies.

For the contour solvers, the greatest challenge is finding a way to choose a suitable number of quadrature points, as we already explained at the end of Chapter 3, together with other possible directions for future research. Interestingly, this question is the main spark behind the birth of the upcoming chapter, where we show how tropical algebra could be a further step towards the solution to this problem.

5 TROPICAL ROOTS OF TROPICAL LAURENT SERIES

Tropical algebra refers to a branch of mathematics developed in the second half of the twentieth century. The adjective *tropical* was given in honour of the Brazilian mathematician Imre Simon: in 1978 he introduced a min-plus structure on $\mathbb{N} \cup \{\infty\}$ to be used in automata theory [Sim78]. In the following decades, researchers started using the nomenclature we are now familiar with. For instance, Cuninghame-Green and Meijer introduced the term *max-algebra* in 1980 [CM80], while *max-plus* appeared in the nineties and early two-thousands [Bac+92; McE06]. We direct the interested reader to Sharify's Doctorate thesis for a deeper introduction and further references on this matter [Sha11].

Even though tropical algebra has risen in many branches of mathematics, such as optimisation [But10], optimal control [AGL08], and tropical geometry [RST05], we are mainly interested in its applications to locate the roots of polynomials or, more in general, the eigenvalues of a sufficiently smooth function. Indeed, having a cheap initial estimate of these values is often very important for numerical methods: for instance, in Chapter 2 and Chapter 3 we have focused on contour integral algorithms, which heavily rely on knowing in which region to search eigenvalues. In 1996, Bini proposed an algorithm to compute polynomial roots based on the Aberth's method and the Newton polygon, but without the formalities of tropical algebra [Bin96]. In 2008, Betcke introduced a diagonal scaling for a matrix polynomial $P(z)$ to improve the conditioning of its eigenvalues λ_j near a target eigenvalue λ_0, which requires the knowledge of $|\lambda_0|$ [Bet08]. The following year, Gaubert and Sharify showed that the tropical roots of a scalar polynomial can give bounds on its "standard" roots [GS09]. Later, Bini and Noferini used the Ehrlich–Aberth method for the polynomial eigenvalue problem, which once again needed a starting point of the order of the absolute value of the eigenvalues [BN13], while Noferini, Sharify, and Tisseur extended the results in [GS09] to matrix-valued polynomials, and compared their bounds with the ones in [BNS13; Mel13].

The goal of this chapter is the generalisation of those results to the case of meromorphic (matrix-valued) functions expressed as Laurent series. The story behind it is, in a sense, quite simple. We thought that having localisation results for some holomorphic functions would have been quite useful and we started by thinking that the same results of [NST15] should hold as well for Taylor series (minus some technicalities), given that they are "just" polynomials with infinite terms. Then we realised that the theory needed for this generalisation allowed us to consider Laurent series with minor changes. The chapter is structured as follows. In section 5.1 we introduce the notation and the background settings. In section 5.2 we develop the theory needed to generalise tropical polynomials to tropical Laurent series, where plenty of examples will help the reader understand the differences between these two settings. The goal of section 5.3 is showing the relationship between the eigenvalues of a meromorphic function and its tropical roots. In section 5.4 we discuss two possible applications, among which there is the link between the quadrature rules of contour solvers and tropical algebra, which was the initial seed of this work. Finally, Section 5.5 is dedicated to the concluding remarks.

5.1 INTRODUCTION

In this chapter we focus on two isomorphic algebraic structures of tropical algebra, named *max-plus* and *max-times algebra*. The max-plus algebra is the semiring $(\mathbb{R}_{\max}, \oplus, \otimes)$, where $\mathbb{R}_{\max} := \mathbb{R} \cup \{-\infty\}$ and the addition and multiplication are defined as

$$a \oplus b := \max(a, b), \qquad a \otimes b := a + b.$$

The zero and the unit element are $-\infty$ and 0, respectively, with the usual convention that $-\infty + a = -\infty$. Similarly, the max-times algebra is the semiring $(\mathbb{R}_{\max,\times}, \oplus, \otimes)$, where $\mathbb{R}_{\max,\times} := \mathbb{R}^+$ is the set of nonnegative real numbers and the addition and multiplication are defined as

$$a \oplus b := \max(a, b), \qquad a \otimes b := ab.$$

As hinted above, the two semirings are isomorphic through the logarithm mapping,

$$\begin{array}{rcl} \log : \mathbb{R}_{\max,\times} & \mapsto & \mathbb{R}_{\max} \\ x & \to & \log(x) \end{array}$$

with the convention $\log(0) = -\infty$. Hence, the choice of which semiring to use is a matter of personal taste. In this thesis we mainly focus on the max-times semiring, hence when considering elements $a \in \mathbb{R}_{\max}$, we will usually write them as $a = \log b$, with $b \in \mathbb{R}_{\max,\times}$, to underline our preferred point of view.

A max-times *tropical polynomial* of *degree d* is a formal expression

$$\mathsf{t}_\times p(x) = \bigoplus_{j=0}^{d} b_j \otimes x^{\otimes j} = \max_{0 \leqslant j \leqslant d} (b_j x^j),$$

where $b_0, \ldots, b_d \in \mathbb{R}_{\max,\times}$, $b_d \neq 0$, and d is a nonnegative integer. If we look at $\mathsf{t}_\times p(x)$ as a real-valued function, then it is piecewise polynomial. The nonzero *max-times roots* (or *tropical roots*) of $\mathsf{t}_\times p(x)$ are the points where $\mathsf{t}_\times p(x)$ is not differentiable, which correspond to the points where the maximum is attained at least twice. Cuninghame-Green and Meijer proved the tropical version of the fundamental algebra theorem: we can rewrite $\mathsf{t}_\times p(x) = b_d \bigotimes_{j=1}^{d} (x, \widetilde{\alpha}_j) = b_d \prod_{j=1}^{d} \max(z, \widetilde{\alpha}_j)$, where $\widetilde{\alpha}_j$ are the tropical roots of $\mathsf{t}_\times p(x)$ [CM80]. The *multiplicity* of $\widetilde{\alpha}_j$ is the cardinality of the set

$$\{k \in \{1, \ldots, d\} : \widetilde{\alpha}_j = \widetilde{\alpha}_k\},$$

which can be computed as the change of derivative at $\widetilde{\alpha}_j$ of the logarithm of the polynomial, i.e.,

$$\lim_{\varepsilon \to 0} \frac{d \log \mathsf{t}_\times p}{dz}|_{z+\varepsilon} - \frac{d \log \mathsf{t}_\times p}{dz}|_{z-\varepsilon}.$$

From now on, we will denote with $\alpha_1, \dots, \alpha_q$ the distinct roots of $\mathrm{t}_\times p(x)$. If $b_j = 0$ for $j = 0, \dots, s$, then 0 is a tropical root of multiplicity $\inf\{j \mid b_j \neq 0\}$.

Computing the roots of $\mathrm{t}_\times p(x)$ is easy and very cheap: it can be done in $\mathcal{O}(d)$ time. In fact, one can prove that finding the tropical roots is equivalent to building the convex hull of the points $(j, \log b_j)$, which is known as the Newton polygon. We will focus more on this aspect in Section 5.2.1 in the general Laurent case. It follows that researchers are very eager to investigate the relationship between tropical roots and "standard" roots, because this information is practically free. More precisely, given a scalar polynomial $p(z) = \sum_{j=0}^{d} b_j z^j \in \mathbb{C}[z]$, then one defines the *(max-times) tropicalization* of $p(z)$ to be[1]

$$\mathrm{t}_\times p(x) := \max_{0 \leqslant j \leqslant d} (|b_j| x^j).$$

Most of the localisation theorems are based on the following famous result by Rouché, which we report in its general version for meromorphic matrix-valued functions.

Theorem 5.1 (Rouché theorem). *Let $F(z), G(z) \in \mathcal{M}(\Omega_0, \mathbb{C}^{n \times n})$. Assume that $F(z)$ is nonsingular for every $z \in \partial\Omega$. Then, for any matrix norm $\|\cdot\|$ induced by a vector norm on $\mathbb{C}^n$, if $\left\|F^{-1}(z)G(z)\right\| < 1$ for every z on $\partial\Omega$, then $F(z)$ and $F(z) + G(z)$ have the same number of eigenvalues minus poles in Ω.*

The main idea for these bound theorems, at least for scalar polynomials $p(z)$, is decomposing $p(z) = q(z) + s(z)$, for some $q(z), s(z) \in \mathbb{C}[z]$, and then showing that $\left|s^{-1}(z)q(z)\right| < 1$ on concentric circles of radii $r_1 < r_2$ thanks to the tropical roots. If we then know where the roots of $s(z)$ lie, we can draw conclusions on the number of roots of $p(z)$ in

$$\mathring{\mathcal{A}}(r_1, r_2) := \{z : r_1 < |z| < r_2\}.$$

1 Note how we use the variable z for "standard" functions, while x for tropical ones.

There are several propositions of this kind and once again we direct the interested reader to Sharify's thesis [Sha11] and his other works for a better understanding. Among them we are mainly interested in the following result.

Theorem 5.2 ([Sha11, Theorem 3.3.3]). *Let $p(z) = \sum_{j=0}^{d} b_j z^j$ be a polynomial with roots $\zeta_1, \dots, \zeta_d$ ordered by increasing absolute value and let $t_\times p(x)$ be its tropicalization with p distinct tropical roots $\alpha_1 < \alpha_2 < \cdots < \alpha_p$, with multiplicity $m_1, m_2, \dots, m_p$. Define $\delta_1 := \alpha_1/\alpha_2, \dots, \delta_{p-1} = \alpha_{p-1}/\alpha_p$ as the parameters that measure the separation among the tropical roots. Then*

- *$p(z)$ has exactly m_j roots in the annulus $\mathring{\mathcal{A}}\left(\frac{1}{2}\alpha_j, 2\alpha_j\right)$ if:*
  - *for $1 < j < d$, $\delta_{j-1}, \delta_j < (2^{m_j+2} + 2)^{-1}$;*
  - *for $j = 1$, $\delta_1 < (2^{m_1+1} + 2)^{-1}$;*
  - *for $j = p$, $\delta_{p-1} < (2^{m_p+1} + 2)^{-1}$;*
- *$p(z)$ has m_j roots in the annulus $\mathring{\mathcal{A}}\left(\frac{1}{3}\alpha_j, 3\alpha_j\right)$ if:*
  - *for $1 < j < d$, $\delta_{j-1}, \delta_j < \frac{1}{9}$;*
  - *for $j = 1$, $\delta_1 < \frac{1}{9}$;*
  - *for $j = p$, $\delta_{p-1} < \frac{1}{9}$.*

In 2014, Noferini, Sharify and Tisseur generalised the second part of Theorem 5.2 to matrix-valued polynomials $P(z) \in \mathbb{C}^{n\times n}[z]$ [NST15, Theorem 2.7]. Here we take one step further and we show that they extend quite naturally to meromorphic functions expressed as Laurent series.

5.2 FROM TROPICAL POLYNOMIALS TO TROPICAL SERIES

Before diving into the technicalities used to extend the aforementioned result, we have to define precisely what tropical Laurent functions are.

Definition 5.1 (Tropical Laurent series)**.** Let $(b_j)_{j\in\mathbb{Z}}$ be a sequence of elements of $\mathbb{R}^+$, indexed by integers and not all zero. A *max-times tropical Laurent series* is

$$\mathrm{t}_\times f(x) := \sup_{j\in\mathbb{Z}}(b_j x^j),$$

with $x \in \mathbb{R}^+$ and takes values in $\mathbb{R}^+ \cup \{+\infty\}$. Similarly, a *max-plus tropical Laurent series* is a function of variable $x \in \mathbb{R} \cup \{-\infty\}$

$$\mathrm{t}f(x) := \sup_{j\in\mathbb{Z}}(\log b_j + jx),$$

taking values in $\mathbb{R} \cup \{\pm\infty\}$. In addition, let $f(z) = \sum_{j\in\mathbb{Z}} b_j z^j$ be a complex function defined by a Laurent series. Then $\mathrm{t}_\times f(x) = \sup_{j\in\mathbb{Z}}(|b_j| x^j)$ is the *(max-times) tropicalization* of $f(z)$. A similar definition holds for the (max-plus) tropicalization.

Remark 5.1. There are two obvious differences in Definition 5.1 with respect to the polynomial case. First, it is the use of supremum in lieu of the maximum; this is necessary because there might be points x where the value of $\mathrm{t}_\times f(x)$ is not attained by any of the polynomial functions $b_j x^j$. In addition, note that $\mathrm{t}_\times f(x)$ ($\mathrm{t}f(x)$, respectively) is not usually well-defined on the whole $\mathbb{R}_{\max,\times}$ ($\mathbb{R}_{\max}$, respectively) as a real-valued function. The next sections are going to clarify these aspects.

Let $D \subset \mathbb{R}^+$ be the largest domain where $\mathrm{t}_\times f(x)$ is well-defined as a real-valued function. It is not too difficult to prove that D is an interval. Indeed, the next lemma is a well-known result of convex analysis [Roc70, Theorem 5.5].

Lemma 5.3. *Let $g_j(z)$ be a set of real-valued functions all defined and convex on the same interval $\Omega \subseteq \mathbb{R}$, and indexed over some non-empty set $\mathcal{I}$, possibly infinite or even uncountable. Then, the largest domain of definition of the function*

$$g : D \to \mathbb{R}, \qquad x \mapsto g(x) = \sup_{j\in\mathcal{I}} g_j(x)$$

is an interval $D \subseteq \Omega \subseteq \mathbb{R}$; moreover, $g(x)$ is convex on D.

Proposition 5.4. *Let $(b_j)_{j\in\mathbb{Z}}$ be a sequence of elements of $\mathbb{R}^+$, indexed by integers and not all zero. Let moreover $\mathcal{I} := \{j \in \mathbb{Z} : b_j > 0\}$. Then the following statements are true:*

1. *The domain of the function*

$$\mathrm{t}_\times f : D \to \mathbb{R}^+, \qquad x \mapsto \mathrm{t}_\times f(x) = \sup_{j \in \mathcal{I}} b_j x^j$$

is an interval $D \subseteq \mathbb{R}^+$; moreover, $\mathrm{t}_\times f(x)$ is convex on D.

2. *The domain of the function*

$$\mathrm{t} f : D_+ \to \mathbb{R}, \qquad x \mapsto \mathrm{t} f(x) = \sup_{j \in \mathcal{I}} (\log b_j + jx)$$

is an interval $D_+ \subseteq \mathbb{R}$; moreover, $\mathrm{t} f(x)$ is convex on D_+.

Proof. The theses follow easily from Lemma 5.3. In fact:

1. For any $j \in \mathcal{I}$, $b_j x^j$ is convex on $\mathbb{R}^+$ (note $j \in \mathcal{I} \Rightarrow b_j > 0$).
2. For any $j \in \mathcal{I}$, $\log b_j \in \mathbb{R}$. Hence, $\log b_j + jx$ is affine, thus convex, on $\mathbb{R}$. $\square$

Remark 5.2. Proposition 5.4 does not specify the nature of the interval D, which can be open, closed, or semiopen (either side being open/closed): indeed, examples can be constructed for all four cases. It can also happen that $D = \mathbb{R}^+$, that D is empty, or that D is a single point. In addition, if $D = [a, b]$ is the domain of $\mathrm{t}_\times f(x)$, then $D_+ := [\log a, \log b]$ is the domain of $\mathrm{t} f(x)$.

An immediate consequence of Corollary 5.4 is that

$$Y := \{y \in D^o : \mathrm{t}_\times f(x) \text{ is not differentiable at } x = y\} \Rightarrow \#Y \leqslant \aleph_0.$$

In particular, $\mathrm{t}_\times f(x)$ (as any convex function of a real variable) has left and right derivative everywhere in its domain of definition D and it is differentiable almost everywhere in the interior of D, except for at most countably many points at which the left and right derivative differ. On the other hand, if $\mathcal{I}$ contains at least two indices, then the set Y may be not empty. Indeed, this leads us to define the set of tropical roots of $\mathrm{t}_\times f(x)$ as $Y \cup (D \backslash D^o)$. More precisely:

Definition 5.2 (Tropical roots of Laurent series)**.** Consider a max-plus Laurent series

$$\mathrm{t}f(x) := \sup_{j\in\mathbb{Z}}(\log b_j + jx),$$

as a real valued function defined on some interval D_+. A point $\log\alpha \in D_+$ is said to be a *tropical root* of $\mathrm{t}f(x)$ if:

1. either $\mathrm{t}f(x)$ is not differentiable at $\log\alpha$, and in this case the multiplicity of $\log\alpha$ is the size of the jump of the derivative at $\log\alpha$ if $\log\alpha$ is a point of nondifferentiability, i.e., $m := \frac{d}{dx^+}\mathrm{t}f(\log\alpha) - \frac{d}{dx^-}\mathrm{t}f(\log\alpha)$;
2. or $\log\alpha = -\infty$ if $b_j = 0$ for $j \leqslant 0$ and the multiplicity is given by $\inf\{j \mid b_j \neq 0\}$;
3. or $\log\alpha$ is a *finite* endpoint of D_+, and the multiplicity of $\log\alpha$ is $m = \infty$.

Similarly, we consider a max-times Laurent series

$$\mathrm{t}_\times f(x) := \sup_{j\in\mathbb{Z}}(b_j x^j),$$

as a real valued function defined on some interval $D \subset \mathbb{R}_{\max,\times}$. Then, $\alpha \in D$ is a tropical root of $\mathrm{t}_\times f(x)$ with multiplicity m if $\log\alpha$ is a tropical root of $\sup_{j\in\mathbb{Z}}(\log b_j + jx)$ with multiplicity m.

As expected, Definition 5.2 falls back to the polynomial case when $\mathrm{t}_\times f(x)$ is a polynomial: if $\mathrm{t}_\times f(x)$ is a polynomial, then $D = [0,\infty[$, hence the second subcase falls back to 0 being a root of $\mathrm{t}_\times f(x)$, while the third one never happens. In addition notice that the multiplicity of a tropical root may be infinite. This happens if α is an endpoint D and belongs to D, that is, if $\mathrm{t}_\times f(\alpha) \in \mathbb{R}$ but either $\mathrm{t}_\times f(x) = +\infty$ for $x > \alpha$ or $\mathrm{t}_\times f(x) = +\infty$ for $x < \alpha$. Analogous observations can be made for $\mathrm{t}f(x)$.

Before continuing with our theoretical exposition, it is better to get some insights on the different cases of the tropical roots thanks to the upcoming examples.

Example 5.1. *Consider $f(z) = -\log(1-z)$ and its Taylor expansion in $[-1,1[$. Then*

$$f(z) = \sum_{j=1}^{\infty}\frac{z^j}{j} \Rightarrow \mathrm{t}_\times f(x) = \sup_{j\geqslant 1}\left(\frac{x^j}{j}\right) = \begin{cases} x & \text{if } x \leqslant 1; \\ \infty & \text{if } x > 1. \end{cases}$$

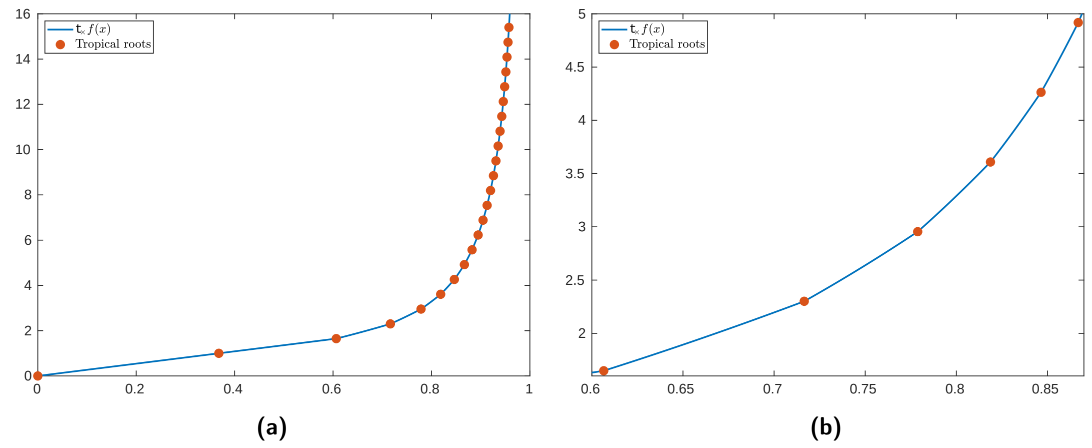


Figure 5.1: The plot of $t_\times f(x)$ (5.1) and its tropical roots α_j in 5.1a and a zoomed in in 5.1b, where we can clearly see that α_j are the points of nondifferentiability.

The domain of $t_\times f(x)$ *is* $D = [0,1]$. *The point* $\alpha_0 = 0$ *is a single root, because* $b_j = 0$ *for* $j \leqslant 0$. *The right endpoint* $\alpha_1 = 1 \in D$ *is a tropical root of infinite multiplicity. They are the only tropical roots because* $t_\times f(x)$ *is differentiable everywhere else in* D. *If we had considered* $g(z) = 1 - \log(1-z)$, *then* $t_\times g(x)$ *would not have had* $\alpha_0 = 0$ *as a tropical root, even though* $D = [0,1]$: *in fact, for a closed endpoint* a *to be a root,* $\log a$ *needs to be finite.*

Example 5.2. *Let* $H_j = \sum_{k=1}^{j} k^{-1}$ *denote the* j*-th harmonic number and consider*

$$f(z) = \sum_{j=1}^{\infty} e^{H_j} z^j \Rightarrow t_\times f(x) = \sup_{j \geqslant 1}(e^{H_j} x^j). \tag{5.1}$$

The domain of $t_\times f(x)$ *is* $D = [0,1[$. *As in the previous example,* $\alpha_0 = 0$ *is a root with multiplicity* 1 *because* $b_j = 0$ *for* $j \leqslant 0$. *The points of nondifferentiability are*

$$\alpha_j = e^{-1/j}, \qquad j = 1,2,3,\dots$$

which have all multiplicity 1, *and accumulate at* 1. *In Figure* 5.1 *we plotted* $t_\times f(x)$ *and* α_j. *Note that* $t_\times f(x) = +\infty$ *if and only if* $x \geqslant 1$. *However,* $1 \notin D$, *so* 1 *itself is not a tropical root.*

Example 5.3. *Let*

$$f(z) = \sum_{j=0}^{\infty} e^{j^2} z^j \Rightarrow t_\times f(x) = \sup_{j \geqslant 0}(e^{j^2} x^j) \equiv +\infty.$$

In this case, the domain of $\mathrm{t}_{\times} f(x)$ *as a real function is empty, and hence there are no tropical roots.*

Remark 5.3. From now on we will assume that 0 is never a tropical root of $\mathrm{t}_{\times} f(x)$. On one hand, this is not a restriction, because if 0 is a root of multiplicity m, then it means $b_j = 0$ for $j \leqslant m$ and thus we can consider a shifted version of $\mathrm{t}_{\times} f(x)$. On the other, it simplifies a lot the exposition, because in this case α is a tropical root only if it is a nondifferentiable point or a closed endpoint of D.

Before discussing the connection between tropical Laurent series $\mathrm{t}_{\times} f(x)$ and the roots of the related function $f(z)$ (or the eigenvalues of $F(z)$), we shall see how the relation between tropical roots and the associated Newton polygon extends from the polynomial setting to the Laurent series case. In addition, in Example 5.2 we witnessed that the tropical roots $\mathrm{e}^{-1/j}$ converge to 1, which is the right endpoint of D, in a similar way to our definition of a tropical root with infinite multiplicity. This does not happen by chance, and it will be the second topic of the next section.

5.2.1 Asymptotic behaviour of tropical roots and infinite Newton polygons

In the previous section, we gave a proper definition for the tropical roots in the Laurent case and we saw that when $\mathrm{t}_{\times} f(x)$ is defined as a real function only on an interval of finite length, then the extrema of this interval are tropical roots of infinite multiplicity. The goals of the upcoming paragraphs are multiple. First, we will show that all the tropical roots are isolated, except for two possible accumulation points. Then we prove a result that mirrors Definition 5.2: the image of $\mathrm{t}_{\times} f(x)$ is contained in $\mathbb{R}^+$ if $x \in]\alpha_{-\infty}, \alpha_{+\infty}[$, under a proper definition of $\alpha_{\pm\infty}$. Finally, we will define the Newton polygon, which can be used to compute the tropical roots with their multiplicities. Clearly, its standard definition for tropical polynomials (see, for instance, [NST15]) extends naturally to the Laurent case, however it requires few more technicalities: some properties are less obvious, since we are dealing with a (possibly) countable number of roots.

Lemma 5.5. *Let* $\mathrm{t}_{\times}f(x)$ *be a max-times Laurent series with tropical roots* $(\alpha_k)_{k\in\mathcal{T}}$ *indexed by* $\mathcal{T}$*. If there are infinitely many roots with positive and/or negative indices that are non-differentiable points, then they are all isolated, with the only possible accumulation points being* $\lim_{k\to\pm\infty}\alpha_k$*.*

Proof. Observe that we can write:

$$\mathrm{t}_{\times}f(x) := \sup_{j\geqslant 0} g_j(x), \qquad g_j(x) := \max_{\substack{k\in\mathbb{Z}\\ |k|\leqslant j}}(b_k x^k). \tag{5.2}$$

Hence, $\mathrm{t}_{\times}f(x)$ can be approximated from below by the functions $g_j(x)$. We consider the non-differentiable points of $g_j(x)$ and see how they change as we let $j\to\infty$. It is clear that to show that the roots are isolated, it suffices to consider the tropical roots of $\mathrm{t}_{\times}f(x)$ that are its non-differentiable points (hence excluding the extrema of the domain D). Therefore we consider the non-differentiable points of $g_j(x)$ and see how they change as we let $j\to\infty$.

First, note that $g_0(x)$ is the constant function $g_0(x)\equiv b_0$. In general, we obtain $g_j(x)$ from $g_{j-1}(x)$ by adding two monomial functions to the set where the supremum is taken. More precisely, these functions are added if and only if $b_j\neq 0$ and $b_{-j}\neq 0$, respectively. Given that the functions added to the supremum have a larger exponent in absolute value than all the others considered before, this introduces at most two new non-differentiable points, which are the leftmost one and the rightmost one. For sake of simplicity, we consider only what happens at the right side of our domain. There are two cases: either the new non-differentiable point, say α, is already the rightmost one, or it superposes or lies on the left of one or more previous tropical roots. In the first scenario, a new tropical root α with multiplicity one is created; in the latter, every tropical root larger than α disappears, and α becomes the rightmost tropical root with multiplicity larger than one. In Figure 5.2 we plotted these two cases while building $g_2(x)$: in 5.2a the new point is already the rightmost one, while in 5.2b superposes with the previous root.

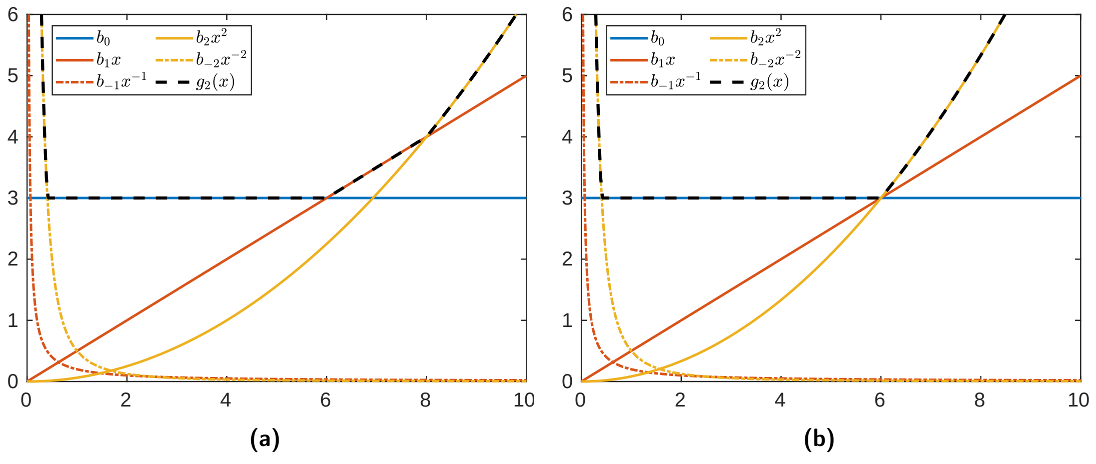


Figure 5.2: Examples of possible scenarios for $g_2(x)$ (black dashed line). In 5.2a the new non differentiable point is the rightmost one, hence it becomes a distinct tropical root with multiplicity one. In 5.2b it superposes with a previous one, hence the multiplicity of the rightmost tropical root is larger than one.

Iterating this process, at most two roots are added at each iteration, of which one is not larger than all the others and one is not smaller than all the others. Hence, there are at most two accumulation points, that are indeed $\lim_{k\to\pm\infty} \alpha_k$. $\square$

Lemma 5.5 guarantees that $\mathcal{T} \subset \mathbb{Z}$ and it makes sense to ask ourselves what is the behaviour of $\lim_{k\to\pm\infty} \alpha_k$. The nature of $\mathcal{T}$ may vary a lot. It may be either bounded or unbounded above, and either bounded or unbounded below, depending on whether the sequence of tropical roots is bi-infinite, infinite on the left, infinite on the right, or finite. For example, if $\mathrm{t}_\times f(x)$ is a polynomial, then $\mathcal{T}$ is clearly finite. The next proposition proves that $\mathrm{t}_\times f(x)$ being a (shifted) polynomial is the only possibility for it to have a finite number of tropical roots with finite multiplicities.

Proposition 5.6. *Consider $\mathrm{t}_\times f(x) = \sup_{j\in\mathbb{Z}} b_j x^j$ and let $\mathcal{I} := \{j \in \mathbb{Z} : b_j > 0\}$. Further, let $(\alpha_k)_{k\in\mathcal{T}}$ be the tropical roots, with $\mathcal{T} \subset \mathbb{Z}$. Then the set $\mathcal{I}$ is bounded above if and only if $\mathcal{T}$ is bounded above and D is not closed on the right. Similarly, $\mathcal{I}$ is bounded below if and only if $\mathcal{T}$ is bounded below and D is not closed on the left.*

Proof. We only prove the case for $\mathcal{I}$ bounded above, being the other one very similar. If $\mathcal{I}$ is bounded above, i.e., $b_j = 0$ definitely for j large enough, then $\mathcal{T}$ is bounded above and $\mathrm{t}_\times f(x) \in \mathbb{R}$ for every x large enough, because

$$\mathrm{t}_\times f(x) = \sup_{\substack{j\in\mathcal{I}\\ j>0}} b_j x^j = \max_{\substack{j\in\mathcal{I}\\ j>0}} b_j x^j.$$

Conversely, assume that $\mathcal{T}$ is bounded above and that D is not closed on the right, i.e., there is no tropical root with infinite multiplicity. Let α_p be the rightmost tropical root and let $j_p \in \mathcal{I}$ be the index where the value $\mathrm{t}_\times f(\alpha_p)$ is attained for the last time. Then, by definition of tropical root, for every $x > \alpha_p$ it holds

$$b_{j_p} x^{j_p} \geqslant b_j x^j,$$

for every $j > 0$, which implies that $\mathrm{t}_\times f(x) \in \mathbb{R}$ for x large enough. Equivalently, we can rewrite the previous equation as

$$x^{j-j_p} \leqslant \frac{b_{j_p}}{b_j}. \tag{5.3}$$

Now assume that $\mathcal{I}$ is not bounded above. Then there exist infinite values of j such that $b_j > 0$ and $j > j_p$. This yields a contradiction, because $\lim_{x\to\infty} x^{j-j_p} = \infty$, but Equation (5.3) implies this value is bounded above by a constant for every $x \geqslant \alpha_p$. □

We can now give a proper definition of $\alpha_{\pm\infty}$, which covers all the cases of $\mathcal{T}$. Note that if $\mathcal{T}$ is bounded above then $\sup\mathcal{T} \in \mathcal{T}$ and if $\mathcal{T}$ is bounded below then $\inf\mathcal{T} \in \mathcal{T}$.

Definition 5.3. Given a sequence $(\alpha_k)_{k\in\mathcal{T}}$ of tropical roots and the set $\mathcal{T}$ of the indices of such sequence, denote $p = \sup\mathcal{T}$ and $f = \inf\mathcal{T}$. Then, $\alpha_{\pm\infty}$ are elements of $\mathbb{R}^+ \cup \{+\infty\}$ defined as follows.

$$\alpha_{+\infty} = \begin{cases} +\infty & \text{if } p < +\infty \text{ and } \alpha_p \text{ has finite multiplicity;} \\ \alpha_p & \text{if } p < +\infty \text{ and } \alpha_p \text{ has infinite multiplicity;} \\ \lim_{k\to+\infty} \alpha_k & \text{if } p = +\infty. \end{cases}$$

$$\alpha_{-\infty} = \begin{cases} 0 & \text{if } f > -\infty \text{ and } \alpha_f \text{ has finite multiplicity;} \\ \alpha_f & \text{if } f > -\infty \text{ and } \alpha_f \text{ has infinite multiplicity;} \\ \lim_{k \to -\infty} \alpha_k & \text{if } f = -\infty. \end{cases}$$

We saw that if $t_\times f(x)$ is well defined as a real-valued function on $D = [a, b]$, then a and b are tropical roots of infinite multiplicity. A priori, $t_\times f(x)$ may also have infinite roots and it may value $\lim_{k\to+\infty} \alpha_k = b$: indeed, Lemma 5.5 does not exclude this scenario. However, we will see that this cannot happen, and if $t_\times f(x)$ has infinite roots, then they are either unbounded, or their limit is a real number outside D (see Example 5.2).

Before proving the main result of this section, we need to define the Newton polygon for Laurent series and show that the usual result linking the tropical roots with the slopes of the Newton polygon holds true for the isolated ones.

Definition 5.4 (Newton polygon for isolated roots)**.** Let $t_\times f(x) = \sup_{j\in\mathbb{Z}} b_j z^j$ be a max-times tropical Laurent series. We define the Newton polygon $\mathcal{N}_{t_\times f}$ as the upper convex hull of $\{(j, \log b_j) \ : b_j \neq 0\}_{j\in\mathbb{Z}}$ and we denote with

$$\mathcal{K}_{t_\times f} := \{j \mid (j, \log b_j) \in \mathcal{N}_{t_\times f}\}$$

the vertices of the Newton polygon for $t_\times f(z)$.

Lemma 5.7 (Newton Polygon for isolated roots)**.** *Let $t_\times f(x)$ be a tropical Laurent series, and $\mathcal{N}_{t_\times f}$ its Newton polygon. Then for each segment connecting two vertices $(j_k, \log b_{j_k})$ and $(j_{k+1}, \log b_{j_{k+1}})$ with $j_k \neq j_{k+1}$ we have a tropical root α_k corresponding to a point of non-differentiability, with finite multiplicity m_k, where:*

$$\alpha_k = \left(\frac{b_{j_k}}{b_{j_{k+1}}}\right)^{\frac{1}{j_{k+1}-j_k}}, \qquad m_k := j_{k+1} - j_k.$$

Alternatively, the isolated roots of $t_\times f(x)$ are the exponential of the opposite of the slopes of the corresponding segments.

Proof. By Lemma 5.5, all the tropical roots α_k who are non-differentiable points are isolated, hence there exists $\varepsilon > 0$ such that:

$$t_\times f(x) = \begin{cases} b_{j_k} x^{j_k} & \alpha_k - \varepsilon \leqslant x \leqslant \alpha_k \\ b_{j_{k+1}} x^{j_{k+1}} & \alpha_k \leqslant x \leqslant \alpha_k + \varepsilon \end{cases}$$

for some $j_k \leqslant j_{k+1}$. By equating the two expressions at α_k we obtain the desired result:

$$\alpha_k = \left(\frac{b_{j_k}}{b_{j_{k+1}}} \right)^{\frac{1}{j_{k+1}-j_k}}, \qquad m_k = j_{k+1} - j_k. \qquad \square$$

Theorem 5.8. *Let $t_\times f(x)$ be a tropical Laurent series, and $(\alpha_k)_{k \in \mathcal{T}}$ be the sequence of tropical roots. Let $\alpha_{\pm\infty}$ be defined as in Definition 5.3 Then,*

$$x \in]\alpha_{-\infty}, \alpha_{+\infty}[\Rightarrow t_\times f(x) \in \mathbb{R}^+$$

and

$$x \notin [\alpha_{-\infty}, \alpha_{+\infty}] \Rightarrow t_\times f(x) = \infty.$$

Furthermore, $t_\times f(\alpha_{+\infty}) \in \mathbb{R}^+$ if and only if p is finite and α_p has infinite multiplicity (respectively, $t_\times f(\alpha_{-\infty})$, f and α_f).

Proof. We only prove the claim for $\alpha_{+\infty}$, i.e., we argue that $t_\times f(x)$ is finite if $x < \alpha_{+\infty}$ and large enough, and that $t_\times f(x)$ is infinite if $x > \alpha_{+\infty}$; the claim for $\alpha_{-\infty}$ admits an identical proof, so we omit it. We analyse the three possible cases in the definition of $\alpha_{+\infty}$.

1. *There is a largest tropical root and it has finite multiplicity.* Note that $\alpha_{+\infty} = +\infty$, hence the second part of the result is vacuously true. The fact that $t_\times f(x) \in \mathbb{R}^+$ for x large enough follows immediately from Proposition 5.6, because $b_j = 0$ for $j > 0$ large enough. Here we have $t_\times f(\alpha_{+\infty}) = +\infty$ by convention.

2. *There is a largest tropical root with infinite multiplicity.* In this case the thesis of the theorem coincides with Definition 5.2, because a root has infinite multiplicity if

and only if is a closed endpoint of D, where $t_\times f(x)$ is a well-defined function. Hence $t_\times f(\alpha_{+\infty}) \in \mathbb{R}^+$.

3. *There is no largest tropical root.* First, $\alpha_{+\infty}$ is well defined because any infinite increasing sequence has a (possibly infinite) limit. There are two subcases.

   - If $\alpha_{+\infty} = +\infty$, then the set of tropical roots is unbounded above. Let $x \in \mathbb{R}^+$ be sufficiently large, then this implies that there exists k such that $\alpha_{k-1} \leqslant x \leqslant \alpha_k$, and therefore

   $$0 < b_{j_k} x^{j_{k-1}} \leqslant t_\times f(x) \leqslant b_{j_k} x^{j_k} < +\infty,$$

   while the second implication is vacuously true. Similarly to the first case, $t_\times f(\alpha_{+\infty}) = +\infty$ by convention.

   - Assume now $\alpha_{+\infty} \in \mathbb{R}^+$. If $x = \alpha_\infty - \varepsilon$ for $\varepsilon > 0$ and not too large, then there exist two tropical roots $\alpha_{k-1} < x < \alpha_k$ and we can conclude that $t_\times f(x) \in \mathbb{R}^+$ arguing similarly to the first subcase; otherwise, if $x \geqslant \alpha_{+\infty}$, write $x = \alpha_{+\infty}(1+\varepsilon)$, with $\varepsilon \geqslant 0$ and define $c_j := b_j \alpha_{+\infty}^j$. Observe

   $$t_\times f(x) \geqslant b_j x^j = b_j(\alpha_{+\infty}(1+\varepsilon))^j = c_j(1+\varepsilon)^j$$

   is true for every j.Since the sequence of tropical roots is increasing, its limit for the index tending to $+\infty$ is an upper bound. Hence, by Lemma 5.7 we have that for all k such that α_k exists

   $$\left(\frac{b_{j_k}}{b_{j_{k+1}}}\right)^{\frac{1}{j_{k+1}-j_k}} < \alpha_{+\infty} \Leftrightarrow c_{j_k} < c_{j_{k+1}}.$$

   Hence, c_{j_k} is also an increasing sequence, and in particular $c_{j_k} > c_{j_\ell}$ for all $k \geqslant \ell$. As a consequence, fixing any ℓ such that α_ℓ is defined,

   $$t_\times f(x) \geqslant b_{j_k} x^{j_k} > c_{j_\ell}(1+\varepsilon)^{j_k} \to \infty \qquad \text{for } k \to +\infty.$$

   The case $\varepsilon = 0$ proves $t_\times f(\alpha_{+\infty}) = +\infty$. □

Theorem 5.9 (Newton polygon for Laurent tropical series)**.** *Let $\mathrm{t}_\times f(x)$ be a tropical Laurent series and let $\mathcal{N}_{\mathrm{t}_\times f}$ be its Newton polygon. Then $\mathrm{t}_\times f(x)$ has a largest finite tropical root α_p of infinite multiplicity if and only if $\mathcal{N}_{\mathrm{t}_\times f}$ has a rightmost segment of infinite length and vertex $(j_p, \log b_{j_p})$, with*

$$\log \alpha_p = \inf_{i \in \mathcal{I}} \sup_{j \geqslant i} \left(-\frac{\log b_j - \log b_i}{j - i} \right)$$

Similarly, $\mathrm{t}_\times f(x)$ has a smallest finite tropical root α_f of infinite multiplicity if and only if $\mathcal{N}_{\mathrm{t}_\times f}$ has a leftmost segment of infinite length and vertex (j_f, b_{j_f}), with

$$\log \alpha_f = \sup_{i \in \mathcal{I}} \inf_{j \leqslant i} \left(-\frac{\log b_i - \log b_j}{i - j} \right).$$

Proof. We only prove the case of the largest tropical root. Assume $\mathrm{t}_\times f(x)$ has a largest tropical root of infinite multiplicity α_p. By Definition 5.2 this corresponds to D being bounded above and closed at its upper endpoint. In addition, the set of indices $\mathcal{I} := \{j : b_j > 0\}$ is not bounded above thanks to Proposition 5.6, while the sequence $\mathcal{T}$ of the indices of the tropical root is bounded above by Theorem 5.8: in particular, there exists a rightmost point of non-differentiability, say, α_{p-1}, which is in correspondence to a segment on the Newton polygon by Lemma 5.7. We label its rightmost segment by $(j_p, \log b_{j_p})$. Then the rightmost convex hull of the points $(j, \log b_j)$ has a rightmost infinite segment. It follows from the definition of convex hull that the slope of this rightmost segment is equal to the supremum of all the slopes of the segments through $(j, \log b_j)$ and $(j_p, \log b_{j_p})$, where the supremum is taken over all the (infinitely many) values in $\{j > j_p\} \cap \mathcal{I}$. Moreover, for any $j_p < i < j$, it must be

$$\frac{\log b_j - \log b_{j_p}}{j - j_p} \leqslant \frac{\log b_j - \log b_i}{j - i},$$

as otherwise α_{p-1} is not the penultimate tropical root. Thus, the opposite of the rightmost slope is

$$\inf_{j \geqslant j_p} \left(-\frac{\log b_j - \log b_{j_p}}{j - j_p} \right) = \inf_{i} \sup_{j \geqslant i} \left(-\frac{\log b_j - \log b_i}{j - i} \right) =: \log \alpha_p. \qquad \square$$

Even though the previous results have been a bit technical, we have finally all the tools to prove the localisation theorems of meromorphic functions thanks to the tropical roots.

5.3 TROPICALIZATION OF ANALYTIC FUNCTIONS

The goal of this section is finding relationships similar to the ones of Theorem 5.2 and their generalisations [NST15, Theorems 2.7-2.8] for eigenvalues of meromorphic Laurent series. More precisely, given the Laurent series

$$F(z) := \sum_{j\in\mathbb{Z}} B_j z^j, \qquad B_j \in \mathbb{C}^{n\times n},$$

we consider $F(z)\colon \Omega \to \mathbb{C}^{n\times n}$, where Ω is the largest annulus where the sum defining $F(z)$ is convergent and thus is holomorphic. We denote this set as $\Omega := \mathring{\mathcal{A}}(R_1, R_2)$, and the radii R_1 and R_2 are determined by the decay rate of $\|B_j\|$ for $j \to \pm\infty$:

$$R_2^{-1} = \limsup_{j\to\infty} \left\|B_j\right\|^{1/j}, \qquad R_1 = \limsup_{j\to\infty} \left\|B_{-j}\right\|^{1/j}, \tag{5.4}$$

where we employ the usual convention $\infty^{-1} = 0$. This definition includes

- polynomials, for which $R_1 = 0$ and $R_2 = \infty$;
- Taylor series, for which $R_1 = 0$ and R_2 is the radius of convergence;
- meromorphic functions in $\mathcal{D}(R_2)$, for which there is only a finite number of negative indices.

The main theorem will hold for meromorphic functions, but most of the results are more general, so we do not require $F(z) \in \mathcal{M}(\mathcal{D}(R_2), \mathbb{C}^{n\times n})$ yet. As in the previous chapters, to avoid the scenario where there are uncountably infinite eigenvalues, we assume $\det F(z) \not\equiv 0$. From now on, to the Laurent series $F(z)$, we associate the tropicalization 5.1

$$\mathrm{t}_\times F(x) := \sup_{j \in \mathbb{Z}} (\|B_j\| x^j) \tag{5.5}$$

where $\|\cdot\|$ is any subordinate matrix norm. In addition, we denote with

$$\delta_j = \frac{\alpha_j}{\alpha_{j+1}} < 1, \qquad \text{for } j \in \mathbb{Z}, \tag{5.6}$$

the ratio between two consecutive tropical roots. Intuitively, one can visualise δ_j on the associated Newton polygon: the smaller δ_j is, the spikier the polygon is in $(k_j, \log B_{k_j})$. For sake of exposition, we consider a simple scalar example.

Example 5.4. *Consider the function*

$$f(z) = \frac{15}{(1-3z)(z-2)},$$

which is holomorphic in $\mathring{\mathcal{A}}(1/3, 2)$. *It is easy to see that*

$$\begin{aligned} f(z) &= 6(\cdots + \frac{1}{3z^2} + \frac{1}{z} + \frac{1}{2} + \frac{z}{4} + \cdots) \\ &= \sum_{j \in \mathbb{Z}} b_j z^j \end{aligned}$$

where $b_j = 2 \cdot 3^{j+2}$ *if* $j < 0$, *and* $b_j = 3 \cdot 2^{-j}$ *for* $j \geqslant 0$. *The (truncated) Newton polygon is given in Figure 5.3 and we can easily detect that there are only the two roots* $\alpha_{-\infty} = 1/3$ *and* $\alpha_\infty = 2$. *In addition, one can check that* $\mathrm{t}_\times f(\alpha_{+\infty}) = 3$ *and* $\mathrm{t}_\times f(\alpha_{-\infty}) = 18$, *hence* $\mathrm{t}_\times f(x)$ *is a well-defined real-valued function in the domain* $D = [\alpha_{-\infty}, \alpha_\infty]$, *in concordance with both Definition 5.2 and Theorem 5.8.*

Example 5.5. *We consider again the function*

$$f(z) = \sum_{j=1}^{\infty} \mathrm{e}^{H_j} z^j \Rightarrow \mathrm{t}_\times f(x) = \sup_{j \geqslant 0} (\mathrm{e}^{H_j} x^j).$$

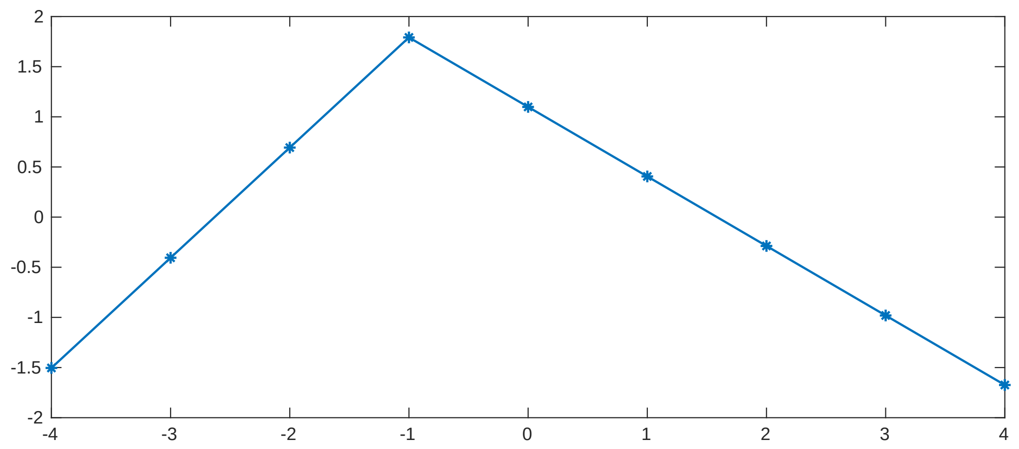


Figure 5.3: The Newton polygon of $\mathrm{t}_\times f(x)$. The two tropical roots $\alpha_{\pm\infty}$ are clearly visible.

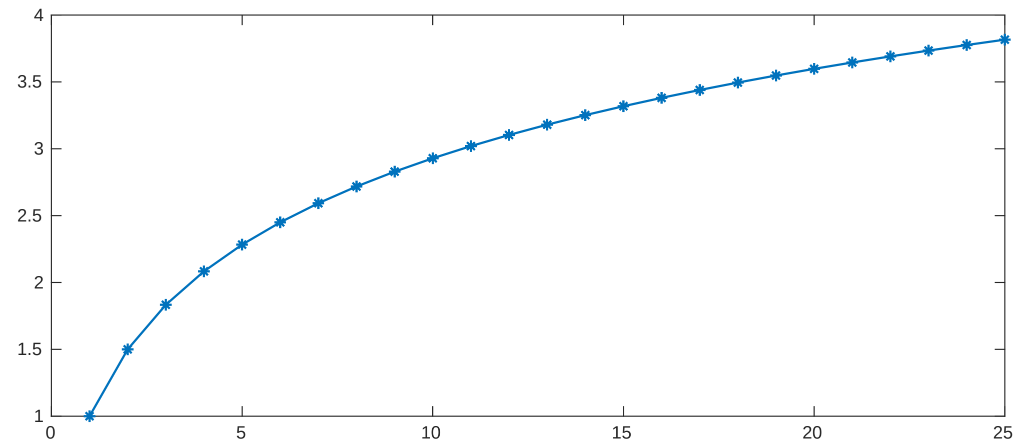


Figure 5.4: The (truncated) Newton polygon of $\mathrm{t}_\times f(x)$. The slopes converge from below to 0.

defined in Example 5.2. We have that the domain of $f(z)$ is the open disk $\Omega = \mathcal{D}(1)$, while the domain D of $\mathrm{t}_\times f(x)$ is $D = [0, 1[$. In Figure 5.4 we plotted its truncated Newton polygon. There we can see that the slopes converge from below to 0, in concordance with the fact that the nonzero roots are $\alpha_j = \mathrm{e}^{-1/j} \to 1$, as j goes to infinity.

In both the previous examples, the tropical roots not only converge to (or are) the endpoints of the domain D of $\mathrm{t}_\times f(x)$, but also to the radii of convergence of $f(z)$. Once again, this does not happen by chance, as explained in the following theorem.

Theorem 5.10. *Consider the Laurent matrix-valued function $F(z) = \sum_{j\in\mathbb{Z}} B_i z^j$, holomorphic in the open annulus $\Omega = \mathring{\mathcal{A}}(R_1, R_2)$, where R_1 and R_2 are defined in (5.4). Let $(\alpha_k)_{k\in\mathcal{T}}$ be the sequence of distinct tropical roots of the tropicalization $\mathrm{t}_\times F(x)$ and let $\alpha_{\pm\infty}$ be the quantities defined in 5.3. Then $R_2 = \alpha_{+\infty}$ and $R_1 = \alpha_{-\infty}$.*

Proof. The proof flows similarly to the one of Theorem 5.8. We prove the statement only for R_2, because the one for R_1 is identical.

1. *There is a largest tropical root and it has finite multiplicity.* We have $\alpha_{+\infty} = +\infty$, and from Proposition 5.6 it follows $b_j = 0$ for $j > 0$ large enough. Hence $R_2 = +\infty$.

2. *There is a largest tropical root with infinite multiplicity.* Assume that $\alpha_p = S_2$. By definition of tropical roots it holds

$$b_j \alpha_p^j \leqslant b_{j_p} \alpha_p^{j_p}$$

therefore $b_j \leqslant S_2^{j_p - j} b_{j_p}$. This implies that $F(z)$ is well defined for every $|z| < S_2$ by (5.4), hence $S_2 \leqslant R_2$. We now claim that $S_2 \geqslant R_2$. Let now $(b_{j_k})_{k \in \mathbb{N}}$ be the subsequence of $(b_j)_{j \in \mathbb{N}}$ obtained by only keeping the indices corresponding to a curve that attains the supremum in the definition of $t_\times F(x)$. Then

$$\frac{1}{R_2} \geqslant \limsup_{j \to \infty} |b_j|^{1/j} \geqslant \limsup_{k \to \infty} b_{j_k}^{1/j_k}.$$

Let $j_{k'} = p$ be the index corresponding to the last tropical root with infinite multiplicity α_p. Then for all $k > k'$ we have $x^{j_k} b_{j_k} \geqslant x^p b_p$, for every $x \geqslant \alpha_p$ by definition of tropical root. Hence,

$$b_{j_k} \geqslant b_p S_2^{p - j_k} \Rightarrow b_{j_k}^{1/j_k} \geqslant \frac{(S_2^p b_p^{1/j_k})}{S_2}.$$

It follows that

$$\frac{1}{R_2} \geqslant \limsup_{k \to \infty} \frac{(S_2^p b_p)^{1/j_k}}{S_2} = \frac{1}{S_2}.$$

3. *There is no largest tropical root.* By Definition 5.3, $\alpha_{+\infty} = \lim_{j \to +\infty} \alpha_j$. On one hand, if $\alpha_{+\infty} \in \mathbb{R}^+$, then an identical proof to the second point will show that $R_2 = \alpha_{+\infty}$. On the other hand, if $\alpha_{+\infty} = +\infty$, then the sequence of tropical roots is unbounded. But if it is unbounded, then (again by the proof of the previous

item) given any tropical root α_k the power series defining $F(z)$ must converge for all $|z| < e^{\alpha_k}$, hence $R_2 = +\infty$. □

5.3.1 Eigenvalue localisation for Laurent series

In this section we generalise the results presented by Noferini, Sharify, and Tisseur in 2015 [NST15] concerning the localisation of the eigenvalues of matrix-valued polynomials. Since we are going to follow their footsteps, we will only report the results that do not literally change, while proving the ones where some little adjustments are needed.

First, we consider a tropical root α_j of $\mathrm{t}_\times F(x)$. Notice that it does not make sense for α_j to be a root with infinite multiplicity, because $F(z)$ would not be defined there. Hence α_j is a point of non differentiability. We use it as a parameter scaling, with $\mu := \alpha_j^{-1} z$ and $\widetilde{F}(\mu)$ being

$$\widetilde{F}(\mu) := \left(\mathrm{t}_\times F(\alpha_j)\right)^{-1} F(z) = \left(\left\|B_{k_{j-1}}\right\|_2 \alpha_j^{k_{j-1}}\right)^{-1} F(\alpha_j \mu) = \sum_{i \in \mathbb{Z}} \widetilde{B}_i \mu^i, \tag{5.7}$$

where

$$\widetilde{B}_i = \left(\left\|B_{k_{j-1}}\right\|_2 \alpha_j^{k_{j-1}}\right)^{-1} B_i \alpha_j^i. \tag{5.8}$$

Lemma 5.11 ([NST15, Lemma 2.2]). *The norms of the coefficients $\widetilde{B}_i$ (5.8) of the scaled function have the following properties:*

$$\left\|\widetilde{B}_i\right\|_2 \leqslant \begin{cases} \delta_{j-1}^{\,k_{j-1}-i} & \text{if} \quad i < k_{j-1}, \\ 1 & \text{if} \quad k_{j-1} < i < k_j, \\ \delta_j^{\,i-k_j} & \text{if} \quad i > k_j, \end{cases} \qquad \left\|\widetilde{B}_{k_{j-1}}\right\|_2 = \left\|\widetilde{B}_{k_j}\right\|_2 = 1.$$

As explained in the introduction of the chapter, our goal is invoking Rouché's Theorem 5.1, therefore we decompose $\widetilde{F}(\mu)$ as the sum of

$$S(\mu) = \sum_{i=k_{j-1}}^{k_j} \widetilde{B}_i\mu^i, \qquad Q(\mu) = \sum_{i\notin\{k_{j-1},\dots,k_j\}} \widetilde{B}_i\mu^i. \tag{5.9}$$

Note that $S(\mu)$ is a matrix-polynomial up to a factor μ^k for some $k \in \mathbb{N}$, while $Q(\mu)$ is generally still a Laurent function. The following two lemmas localise the nonzero eigenvalues of $S(\mu)$.

Lemma 5.12 ([NST15, Lemma 2.3]). *Let $P(z) = \sum_{j=0}^{\ell} B_j z^j$ with $B_0, B_\ell \neq 0$ be a regular matrix polynomial. Then every eigenvalue λ of $P(z)$ satisfies*

$$(1+\kappa(B_0))^{-1}\alpha_1 \leqslant |\lambda| \leqslant (1+\kappa(B_\ell))\,\alpha_q,$$

where α_1, α_q are the smallest and largest finite tropical roots of $\mathrm{t}_\times P(x)$, respectively. Furthermore, if both B_0 and B_ℓ are invertible, the inequalities are strict.

Lemma 5.13 ([NST15, Lemma 2.4]). *Let $m_j := k_j - k_{j-1}$. If $B_{k_{j-1}}$ and B_{k_j} are nonsingular, then the nm_j nonzero eigenvalues of $S(\mu)$ lie in the open annulus $\mathring{\mathcal{A}}((1+\kappa(B_{k_{j-1}}))^{-1}, 1+\kappa(B_{k_j}))$.*

Proof. If $k_{j-1} \geqslant 0$, then the lemma is equivalent to [NST15, Lemma 2.4]. Otherwise, consider the polynomial $\widetilde{S}(\mu) = \mu^{-k_{j-1}}S(\mu)$, which we can apply [NST15, Lemma 2.4] to. It follows that $\widetilde{S}(\mu)$ has nm_j nonzero eigenvalues in $\mathring{\mathcal{A}}((1+\kappa(A_{k_{j-1}}))^{-1}, 1+\kappa(A_{k_j}))$, which yields that $S(\mu)$ has nm_j nonzero eigenvalues and nk_{j-1} poles in zero. □

Lemma 5.14 ([NST15, Lemma 2.5]). *The following inequalities hold for $Q(\mu)$ and $S(\mu)$ in (5.9),*

$$\left\|S(\mu)^{-1}\right\|_2 \leqslant \begin{cases} \dfrac{\kappa(B_{k_j-1})|\mu|^{-k_{j-1}}(1-|\mu|)}{1-|\mu|\big(1+\kappa(B_{k_j-1})(1-|\mu|^{m_j})\big)} & \text{if } 0<|\mu|\leqslant \big(1+\kappa(B_{k_j-1})\big)^{-1}, \\[2ex] \dfrac{\kappa(B_{k_j})|\mu|^{-k_j}(|\mu|-1)}{|\mu|-1-\kappa(B_{k_j})(1-|\mu|^{-m_j})} & \text{if } |\mu|\geqslant 1+\kappa(B_{k_j}). \end{cases},$$

$$\|Q(\mu)\|_2 \leqslant \frac{\delta_{j-1}|\mu|^{k_{j-1}}}{|\mu|-\delta_{j-1}} + \frac{\delta_j|\mu|^{k_j+1}}{1-\delta_j|\mu|} \qquad \text{if } \delta_{j-1}<|\mu|<\frac{1}{\delta_j}.$$

Proof. The case of $\|S(\mu)\|_2$ is identical to [NST15, Lemma 2.5]. The same is true for $\|Q(\mu)\|_2$, but we recall the proof because $Q(\mu)$ is generally a Laurent series. Assume that $\delta_{j-1} < |\mu| < \delta_j^{-1}$. It follows from (5.9) and Lemma 5.11 that

$$\begin{aligned}\|Q(\mu)\|_2 &\leqslant \sum_{i<k_{j-1}-1} \delta_{j-1}{}^{k_{j-1}-i}|\mu|^i + \sum_{i>k_j+1} \delta_j{}^{i-k_j}|\mu|^i \\ &\leqslant \frac{\delta_{j-1}(|\mu|^{k_{j-1}} - \delta_{j-1}{}^{k_{j-1}})}{|\mu| - \delta_{j-1}} + \frac{\delta_j|\mu|^{k_j+1}(1-(\delta_j|\mu|)^{-k_j})}{1-\delta_j|\mu|}\end{aligned}$$

and the bound in the thesis follows since $\delta_{j-1} < |\mu| < \delta_j^{-1}$. □

After the upcoming technical lemma, we will have all the tools to generalise Theorem 5.2 to meromorphic matrix-valued functions. Not every statement translates exactly under these new settings, given the possible presence of the poles, but the ability of locating the eigenvalues of $F(z)$ thanks to the associated tropical roots still holds true.

Lemma 5.15 ([NST15, Lemma 2.6]). *For given $c, \delta > 0$ such that $\delta \leqslant (1+2c)^{-2}$, the quadratic polynomial*

$$p(r) = r^2 - \left(2 + \frac{1-\delta}{\delta(1+c)}\right)r + \frac{1}{\delta}$$

has two real roots

$$f := f(\delta, c) = \frac{(1+2c)\delta + 1 - \sqrt{(1-\delta)(1-(1+2c)^2\delta)}}{2\delta(1+c)}, \qquad g = (\delta f)^{-1},$$

with the properties that

1. $1 < 1 + c \leqslant f \leqslant g$,
2. $\dfrac{1}{f-1} + \dfrac{1}{g-1} = \dfrac{1}{c}$.

Theorem 5.16. *Let $\ell^- > -\infty$ and let $F(\lambda) = \sum_{j=\ell^-}^{\infty} B_j\lambda^j$ be a regular, meromorphic Laurent function, analytic in the open annulus $\Omega := \mathring{\mathcal{A}}(R_1, R_2)$. For every $j \in \mathbb{Z}$, let $f_j = f(\delta_j, \kappa(B_{k_j}))$, where $f(\delta, c)$ is defined as in Lemma 5.15, and $g_j = (\delta_j f_j)^{-1}$. Then, the following statements hold true:*

1. *If $\delta_j \leqslant (1+2\kappa(B_{k_j}))^{-2}$, then $F(\lambda)$ has exactly nk_j eigenvalues minus poles inside the disk $\mathcal{D}((1+2\kappa(B_{k_j}))\alpha_j)$ and it does not have any eigenvalue inside the open annulus $\mathring{\mathcal{A}}((1+2\kappa(B_{k_j}))\alpha_j, (1+2\kappa(B_{k_j}))^{-1}\alpha_{j+1})$.*

2. *For any $j < s$, if $\delta_j \leqslant (1+2\kappa(B_{k_j}))^{-2}$ and $\delta_s \leqslant (1+2\kappa(B_{k_s}))^{-2}$, then $F(\lambda)$ has exactly $n(k_s - k_j)$ eigenvalues inside the closed annulus $\mathcal{A}((1+2\kappa(B_{k_j}))^{-1}\alpha_{j+1}, (1+2\kappa(B_{k_s}))\alpha_s)$.*

Proof. As a preliminary step, note that for a fixed $c \geqslant 1$ and $\delta \leqslant (1+2c)^{-2}$, the function $f(\delta, c)$ of Lemma 5.15 is increasing and attains its maximum, which is $1+2c$, at $\delta = (1+2c)^{-2}$. Therefore it holds $f(\delta_j, \kappa(B_{k_j})) \leqslant 1+2\kappa(B_{k_j})$ for $\delta_j \leqslant (1+2\kappa(B_{k_j}))^{-2}$.

1. We assume that $\delta_j \leqslant (1+2\kappa(B_{k_j}))^{-2}$ and we partition $\widetilde{F}(\mu)$ as in (5.9). Let r be such that

$$1+\kappa(B_{k_j}) < r < \delta_j^{-1}. \tag{5.10}$$

This r is well defined because $\delta_j \leqslant (1+2\kappa(B_{k_j}))^{-2} < (1+\kappa(B_{k_j}))^{-1}$. It follows by Lemma 5.13 that $S(\mu)$ is nonsingular on the circle $\Gamma_r = \{\mu \in \mathbb{C} : |\mu| = r\}$. In order to apply Theorem 5.1 with $\widetilde{F}(\mu) = S(\mu) + Q(\mu)$ and Γ_r, we must check that $\left\|S(\mu)^{-1}Q(\mu)\right\|_2 < 1$ for all $\mu \in \Gamma_r$. Since $|\mu| = r$ with r such that

$$\delta_{j-1} < 1 < 1+\kappa(B_{k_j}) < r < \delta_j^{-1}, \tag{5.11}$$

we can apply the bounds in Lemma 5.14. They yield

$$\begin{aligned}\left\|S(\mu)^{-1}Q(\mu)\right\|_2 &\leqslant \left\|S(\mu)^{-1}\right\|_2 \|Q(\mu)\|_2 \\ &\leqslant \frac{r^{-k_j}(r-1)\kappa(B_{k_j})}{r-1-\kappa(B_{k_j})(1-r^{-m_j})}\left(\frac{\delta_{j-1}r^{k_{j-1}}}{r-\delta_{j-1}} + \frac{\delta_j r^{k_j+1}}{1-\delta_j r}\right).\end{aligned}$$

The latter bound is less than 1 if

$$\frac{\delta_{j-1}r^{-m_j}}{r-\delta_{j-1}} + \frac{\delta_j r}{1-\delta_j r} < \frac{r-1-\kappa(B_{k_j})(1-r^{-m_j})}{(r-1)\kappa(B_{k_j})},$$

or equivalently, if

$$\frac{\delta_j r}{1-\delta_j r} < \frac{r-1-\kappa(B_{k_j})}{(r-1)\kappa(B_{k_j})} + r^{-m_j}\left(\frac{1}{r-1} - \frac{\delta_{j-1}}{r-\delta_{j-1}}\right).$$

Since $r - \delta_{j-1} > \delta_{j-1}(r-1)$, the last inequality holds when $\frac{\delta_j r}{1-\delta_j r} < \frac{r-1-\kappa(B_{k_j})}{(r-1)\kappa(B_{k_j})}$, or equivalently when $p(r) < 0$, where $p(z)$ is the polynomial of Lemma 5.15 with $\delta = \delta_j$ and $c = \kappa(B_{k_j})$. It follows from Lemma 5.15 that $p(r)$ is negative for the values of r such that

$$f_j < r < g_j \tag{5.12}$$

by recalling that f_j and g_j are the two roots of p. Note that, by the same lemma, $f_j \geqslant 1 + \kappa(B_{k_j})$ and $g_j \leqslant (\delta_j)^{-1}$ so (5.12) is sharper than (5.11). In addition, for any $|\mu| = r$, where $r = f_j$ or $r = g_j$ the upper bound for $\left\|S(\mu)^{-1}Q(\mu)\right\|_2$ is equal to 1. Therefore, such μ belongs to the domain of analiticity, and we have $R_1 \leqslant f_j < g_j \leqslant R_2$. By Rouché's Theorem 5.1, $S(\mu)$ and $\widetilde{F}(\mu)$ have the same number of eigenvalues minus poles, i.e., nk_j, inside the disk $\mathcal{D}(r)$ for any r such that $f_j < r < g_j$. This also implies that there are no eigenvalues in the open annulus $\mathring{\mathcal{A}}(f_j, g_j)$, because the number of eigenvalues minus poles must remain constant and poles cannot lie there. The thesis then follows from the scaling $z = \mu\alpha_j$ and the preliminary point.

2. By the proof of the previous point, if $\delta_j \leqslant (1 + 2\kappa(B_{k_j}))^{-2}$ then $F(\lambda)$ has nk_j eigenvalues minus poles inside $D((1+2\kappa(B_{k_j}))\alpha_j)$ and it has no eigenvalues in $\mathring{\mathcal{A}}((1+2\kappa(B_{k_j}))\alpha_j, (1+2\kappa(B_{k_j}))^{-1}\alpha_{j+1})$. A similar statement holds when $\delta_s \leqslant (1+2\kappa(B_{k_s}))^{-2}$. Given that poles cannot lie there, this implies that $F(\lambda)$ has exactly $n(k_s - k_j)$ eigenvalues inside $\mathcal{A}((1+2\kappa(B_{k_j}))^{-1}\alpha_{j+1}, (1+2\kappa(B_{k_s}))\alpha_s)$. $\square$

When there is a finite number of negative (positive, respectively) indices, there is an exclusion (inclusion, respectively) disk centered in zero. This is a generalisation of Lemma 5.12.

Theorem 5.17. *Let $F(z) = \sum_{j=\ell^-}^{\ell^+} B_j z^j$ be a regular Laurent series. If $\ell^- > -\infty$ and B_{ℓ^-} is non singular, then $F(z)$ has $n\ell^-$ eigenvalues minus poles at zero, and the other eigenvalues λ of $F(z)$ satisfy*

$$(1+\kappa(B_{\ell^-}))^{-1}\,\alpha_1 \leqslant |\lambda|$$

Similarly, if $\ell^+ < \infty$ and B_{ℓ^+} is non singular, then all eigenvalues satisfy

$$|\lambda| \leqslant (1+\kappa(B_{\ell^-}))\,\alpha_q,$$

where α_q is the maximum tropical root.

Proof. The proof follows the ideas in [HT03, Lemma 4.1]. We start by the case $\ell^+ < \infty$. By the definition of α_q we have

$$\|B_i\|_2 \leqslant \alpha_q^{\ell^+ - i}\|B_{\ell^+}\|_2, \qquad i \leqslant \ell^+.$$

Now assume by contradiction that there exists an eigenvalue $|\lambda| > (1+\kappa(B_{\ell^-}))\,\alpha_q$. Then, we may consider a normalised eigenvector $\|v\|_2 = 1$, and write

$$\begin{aligned}
\|F(\lambda)v\|_2 &\geqslant \left\|B_{\ell^+}|\lambda|^{\ell^+}v\right\|_2 - \sum_{i<\ell^+}\|B_i\|_2|\lambda|^i \\
&\geqslant |\lambda|^{\ell^+}\left(\left\|B_{\ell^+}^{-1}\right\|_2^{-1} - \sum_{i<k}\|B_i\|_2|\lambda|^{i-\ell^+}\right) \\
&\geqslant |\lambda|^{\ell^+}\left(\left\|B_{\ell^+}^{-1}\right\|_2^{-1} - \sum_{i<k}\|B_{\ell^+}\|_2\alpha_q^{\ell^+-i}|\lambda|^{i-\ell^+}\right) \\
&= |\lambda|^{\ell^+}\left\|B_{\ell^+}^{-1}\right\|_2^{-1}\left(1-\kappa(B_{\ell^+})\sum_{i<k}\left[\frac{\alpha_q}{|\lambda|}\right]^{\ell^+-i}\right) \\
&= |\lambda|^{\ell^+}\left\|B_{\ell^+}^{-1}\right\|_2^{-1}\left(1-\kappa(B_{\ell^+})\frac{\alpha_q}{|\lambda|-\alpha_q}\right) > 0
\end{aligned}$$

where the last inequality follows from the assumption $|\lambda| > (1+\kappa(B_{\ell^-}))\,\alpha_q$. Hence, we have the desired upper bound for $|\lambda|$.

If we have $\ell^- > -\infty$ and B_{ℓ^-} nonsingular, we can rewrite $F(z)$ as

$$F(z) = z^{\ell^-} \widehat{F}(z),$$

where $\widehat{F}(z)$ is a Taylor series (or a polynomial) with $\det \widehat{F}(0) \neq 0$. Hence, we conclude that $F(z)$ has an eigenvalue (or pole, depending on the sign of ℓ^-) of the desired multiplicity. Applying the above reasoning to $F(z^{-1})$ yields the lower bound for the remaining eigenvalues. □

5.3.2 Practical computation of the Newton polygon

Given a tropical Laurent series, computing its Newton polygon may be challenging, as the Graham-Scan algorithm [Gra72] could require an infinite number of comparisons. There are two strategies to overcome this problem. First, if the norm of the coefficients B_j are easy to compute, we show that the finite truncations of the Newton polygon of $\mathrm{t}_\times F(x)$ converge in some sense to the infinite one. Second, we consider the case where the Laurent series is given as an elementary function (for instance, the exponential), of which we alter only a finite number of the coefficients. For instance, in Chapter 1 we saw eigenvalue problems arising from delayed differential equations that are polynomials in z and in e^z [Jar12; JM10].

Theorem 5.18 (Truncation of the Newton polygon). *Let $F\colon \Omega \to \mathbb{C}^{n\times n}$ be a Laurent function with $\Omega = \mathring{\mathcal{A}}(R_1, R_2)$ and let $\mathrm{t}_\times F(x)$ be its tropicalization. In addition, consider any "right-finite" truncation $\mathrm{t}_\times F_d(x) = \bigoplus_{0\leqslant k\leqslant d} \|B_k\|_2 x^k$ and let $C_2 > 0$ be a constant such that $\|B_k\|_2 < C_2 R_2^{-k}$ for any $k > 0$. Then for any two consecutive indices $i, j \in \mathcal{K}_{\mathrm{t}_\times F_d}$ such that $\|B_j\|_2 \geqslant \|B_i\|_2 R_2^{-(j-i)}$, let ℓ_2 be defined as*

$$\ell_2 := \frac{(j-i)\log C_2 + i \log \|B_j\|_2 - j \log \|B_i\|_2}{(j-i)\log R_2 + \log \|B_j\|_2 - \log \|B_i\|_2}$$

and suppose moreover that for all $j+1 \leqslant k \leqslant \ell_2$ it holds

$$\|B_k\|_2 \leqslant \|B_j\|_2^{\frac{k-i}{j-i}} \|B_i\|_2^{-\frac{k-j}{j-i}}. \tag{5.13}$$

Then i, j are also consecutive indices in $\mathcal{K}_{\mathsf{t}_\times F}$.

Similarly, let $\mathsf{t}_\times F_{-d}(x) = \bigoplus_{0 \leqslant k \leqslant d} \|B_{-k}\|_2 x^{-k}$ "left-finite" truncation and let $C_1 > 0$ be a constant such that $\|B_{-k}\|_2 < C_1 R_1^k$ for any $k > 0$. Then for any two consecutive indices $-i, -j \in \mathcal{K}_{\mathsf{t}_\times F_{-d}}$ such that $\|B_{-j}\|_2 \geqslant \|B_{-i}\|_2 R_1^{j-i}$, let ℓ_1 be defined as

$$\ell_1 := \frac{(j-i)\log C_1 + i \log \|B_{-j}\|_2 - j \log \|B_{-i}\|_2}{(i-j)\log R_1 + \log \|B_{-j}\|_2 - \log \|B_{-i}\|_2}$$

and suppose moreover that for all $j+1 \leqslant k \leqslant \ell_1$ it holds

$$\|B_{-k}\|_2 \leqslant \|B_{-j}\|_2^{\frac{k-i}{j-i}} \|B_{-i}\|_2^{-\frac{k-j}{j-i}}.$$

Then $-i$, $-j$ are also consecutive indices in $\mathcal{K}_{\mathsf{t}_\times F}$.

Proof. Since $\log \|B_k\|_2 \leqslant C_2 R_2^{-k}$ for $k > 0$, the points $(k, \log \|B_k\|_2)$ lie below the line $L_1\colon y = \log C_2 - x \log R_2$. In addition, let

$$L_2\colon y = \log \|B_j\|_2 + (x-j)\frac{\log \|B_j\|_2 - \log \|B_i\|_2}{j-i}$$

be the line containing the segment between $(i, \log \|B_i\|_2)$ and $(j, \log \|B_j\|_2)$. Now we want to prove that if this segment belongs to $\mathcal{N}_{\mathsf{t}_\times F_d}$, then it belongs to $\mathcal{N}_{\mathsf{t}_\times F}$. In order to do that, we have to show that all the other points $(k, \log \|B_k\|_2)$ lie below L_2. If $j+1 \leqslant k \leqslant \ell_2$, then this condition is equivalent to (5.13). On the other hand, if $k \geqslant \ell_2$, then we can prove that L_2 lies above L_1, and therefore above $(k, \log \|B_k\|_2)$. We can do this by noting that $L_2(k) - L_1(k) \geqslant L_2(\ell_2) - L_1(\ell_2) = 0$. Indeed, for the second equality we have

$$\begin{aligned}
(j-i)(L_2(\ell_2) - L_1(\ell_2)) &= (j-i)\log \|B_j\|_2 + (\ell_2 - j)(\log \|B_j\|_2 - \log \|B_i\|_2) \\
&\quad + (i-j)\log C + \ell_2 (j-i) \log R_2 \\
&= (i-j)\log C + (j-\ell_2)\log \|B_i\|_2 \\
&\quad + (\ell_2 - i)\log \|B_j\|_2 + \ell_2(j-i)\log R_2 \\
&= (i-j)\log C + j \log \|B_i\|_2 - i \log \|B_j\|_2 \\
&\quad + \ell_2((j-i)\log R_2 + \log \|B_j\|_2 - \log \|B_i\|_2) = 0
\end{aligned}$$

by the definition of ℓ_2. For the first inequality, it is sufficient to prove that the slope of L_2 is larger than the slope of L_1. This is equivalent to

$$-(j-i)\log R_2 \leqslant \log \|B_j\|_2 - \log \|B_i\|_2$$

which is true because $\|B_j\|_2 \geqslant \|B_i\|_2 R_2^{-(j-i)}$. The proof for the left truncation is identical, providing changing the slope of L_1 from $-\log R_2$ to $\log R_1$. $\square$

Consider now the case we are given the tropical roots $(\alpha_j)_{j\in\mathcal{T}}$ of a tropical Laurent series $\mathrm{t}_\times f(x)$, and we want to determine the tropical roots of a modified function $\mathrm{t}_\times g(x) = \mathrm{t}_\times(f(x)+p(x))$, where $p(z)$ is a Laurent polynomial. The standard example we have in mind is $g(z) = \mathrm{e}^z + p(z)$: the Newton polygon of $\mathrm{t}_\times(\mathrm{e}^x)$ is the convex hull of the nodes $(j, -\log j!)$, and the tropical polynomial $\mathrm{t}_\times p(x)$ only modifies a finite number of these nodes. For sake of exposition, we assume that the cardinality of the set $\mathcal{I} \subset \mathbb{Z}$ of the nonzero indices b_j is infinite. The case where $\mathcal{I}$ is finite in one or both directions requires minimal modifications.

Since there is a one-to-one correspondence between the tropical roots and the Newton polygon, our problem is equivalent to update the latter thanks to the Graham scan algorithm [Gra72]. Note that we can focus on the case where $p(z) = \gamma$ is a constant. In fact, the case where $p(z)$ is a monomial is a simple shift to the left or to the right, while a general polynomial is just a finite number of compositions of these updates. To summarise, our goal is, given the set of indices $\mathcal{I}$ of $\mathcal{N}_{\mathrm{t}_\times f(x)}$, constructing the modified set of indices $\widehat{\mathcal{I}}$ of $\mathcal{N}_{\mathrm{t}_\times g(x)}$. There are two ways we can do that.

- It may happen that $\widehat{\mathcal{I}}$ and $\mathcal{I}$ only differ by a finite number of elements, hence we can express $\widehat{\mathcal{I}}$ by listing all of them.
- We get $\widehat{\mathcal{I}}$ from $\mathcal{I}$ by dropping all the indices to the right (resp. to the left) of 0, and then adding 0.

The intuition helps us find a way to proceed:

- If $\log\gamma$ lies below the Newton polygon in 0, we simply stop and do not include $(0, \log\gamma)$ in the nodes. Otherwise, we include 0 in $\widehat{\mathcal{I}}$ and move to the next point.

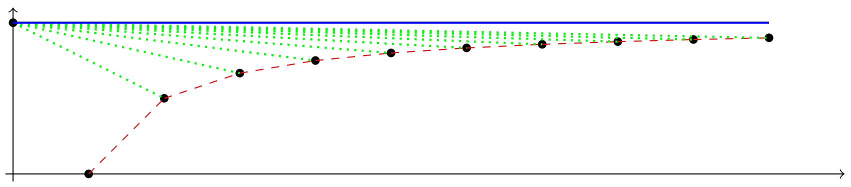


Figure 5.5: Lines considered by the Graham Scan algorithm in Example 5.6. The algorithm does not terminate within a finite number of slope comparisons.

- We start from $i = 1$ and compare the slope of the segment between $(0, \log \gamma)$ and $(j_i, \log b_{j_i})$, with $(j_{i+1}, \log b_{j_{i+1}})$. If the latter slope is smaller, or j_i is the last index in $\mathcal{I}$, then we stop. Otherwise, we remove the index j_i from $\mathcal{I}$, and consider the next indices in the previous point.

The previous paragraphs describe the algorithm for the points on the right of $(0, \log \gamma)$, and the same procedure needs to be repeated for the indices on the left. However, if the cardinality of $\mathcal{I}$ is infinite, we may have to perform an infinite amount of comparisons, as the following example shows.

Example 5.6. *Consider the function*

$$f(z) = \sum_{j=1}^{\infty} \mathrm{e}^{\frac{j-1}{j}} z^j.$$

and its associated tropical series $\mathrm{t}_\times f(x)$. *The nodes of the Newton polygon are* $(j, 1 - j^{-1})$ *and they belong to the upper convex hull, as we show in Figure 5.5 (dashed red line). If we modify this function and consider*

$$g(z) = \mathrm{e} + f(z),$$

then we need to add the node $(0, 1)$ *to the convex hull. If we connect it to* $(j, 1 - j^{-1})$, *we obtain a sequence of lines of increasing slope (dotted green line), that converges to the horizontal line* $y = 1$. *However, the Graham-Scan algorithm does not terminate in a finite number of steps. The final convex hull correspond to* $\{y \leqslant 1\}$, *and is depicted by the blue solid line.*

It seems that we cannot exclude that the Graham scan works in these settings. However, we can prove the case we just described is the only way it can go wrong.

Theorem 5.19. *Let $(j, \log b_j)$ be the nodes of a Newton polygon, $j \in \mathcal{I} \subseteq \mathbb{Z}$; similarly, let $\widehat{\mathcal{I}}$ be the set of indices corresponding to vertices of the Newton polygon for*

$$\widehat{b}_j = \begin{cases} \gamma & \text{if } j = 0 \\ b_j & \text{otherwise} \end{cases}.$$

Then, if any of the following conditions hold, the Graham-scan algorithm applied to the right of 0 terminates within a finite number of steps.

1. $\alpha_\infty = \infty$,
2. $\alpha_\infty < \infty$ *and* $\limsup_{j \in \mathcal{I}} \left[\log(b_j) - j \log \alpha_\infty\right] > \log \gamma$.

In the remaining case, i.e., $\alpha_\infty < \infty$ but the second condition is not satisfied, it follows that $\widehat{\mathcal{I}} \cap \{j \geqslant 0\} = \{0\}$, and the root at zero has infinite multiplcity.

Proof. First, consider the case $\alpha_\infty = \infty$. If there is a finite number of tropical roots, then the thesis is clearly true; otherwise, $\alpha_\infty = \infty$ means that the tropical roots are unbounded. In particular, the Graham scan algorithm compares the slopes defined by the sequences:

$$s_i := \frac{\log b_{j_i} - \log \gamma}{j_i}, \qquad t_i := \frac{\log b_{j_{i+1}} - \log b_j}{j_{i+1} - j_i},$$

and stops as soon as $s_i \geqslant t_i$. We point out that, as long as this condition does not hold, the sequence s_i is increasing, since the "next" node $(j_{i+1}, \log b_{j_{i+1}})$ is above the line passing through $(0, \log \gamma)$ and $(j_i, \log b_{j_i})$. This situation is visible, for instance, in Figure 5.5, where none of the lines satisfies the condition, and hence the slopes keep increasing. In contrast, the sequence t_i converges to $-\infty$. Hence, there exists a finite index i where $s_i \geqslant t_i$, and where the condition is satisfied.

If $\alpha_\infty < \infty$, without loss of generality we can set $\log \alpha_\infty = 0$, and therefore $\alpha_\infty = 1$: we just need to modify the function $f(z)$ by scaling the variable as $f(\alpha_\infty^{-1} z)$, which yields the Laurent coefficients $b_j \alpha_\infty^{-j}$. With this choice, the slope of the segments in the tropical roots converges to 0, and the plot is "asymptotically flat". In addition, all the slopes are non-negative, and therefore the sequence of $\log b_j$ is non-decreasing. We now define $\xi := \lim_{j\to\infty} \log b_j$, and we distinguish two subcases, one where $\xi > \log\gamma$, and the other where $\xi \leqslant \log\gamma$. If $\xi > \log\gamma$, then there exists one point $(j, \log b_j)$ such that the slope of the segment connecting $(0, \log\gamma)$ to it is strictly positive, and this also holds for all the following points, since the b_j are non-decreasing. We now use the same argument as before: the sequences of slopes s_i and t_i are such that s_i is non-decreasing as long as the stopping condition is not satisfied, and $t_i \to 0$. Hence, the algorithm terminates in a finite number of steps. If $\xi \leqslant \log\gamma$, the horizontal line starting from $(0, \log\gamma)$ lies above all other points, but any other line passing through the same point and with negative slope necessarily intersects the previous Newton polygon. Hence, 0 is a tropical roots of infinite multiplicity, and all nodes with positive indices need to be removed. The statement then follows by rephrasing the claim on the original b_j, for a generic $\log\alpha_\infty \neq 0$. □

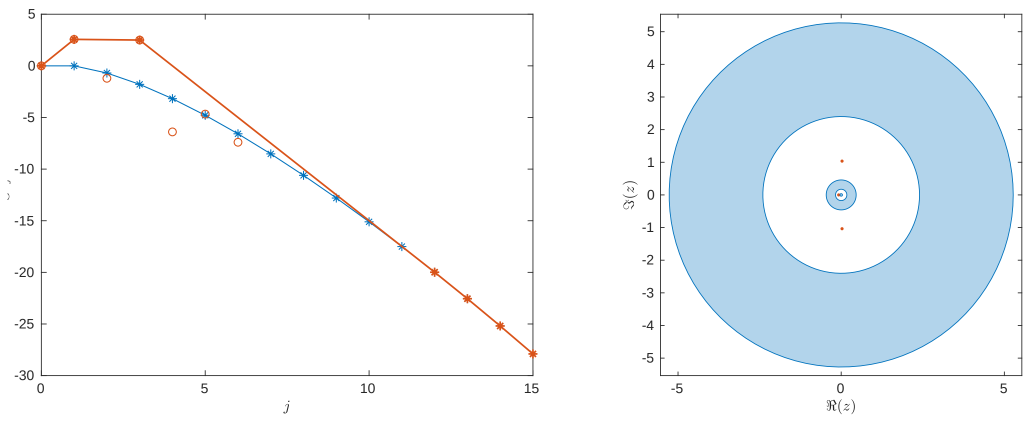
5.4 APPLICATIONS

In this final section we consider two possible applications of our results. We start by considering the problem of finding inclusion sets for the eigenvalues of a Laurent function. We assume that a known series is modified in a finite number of spots and we exploit Theorem 5.19. For sake of simplicity, we work in the scalar scenario.

Example 5.7. *We consider*

$$g(z) = e^z + p(z), \qquad p(z) := 12z - \frac{z^2}{5} + 12z^3 - 0.04z^4 + 10^{-3}z^5 - 0.002z^6.$$

As previously stated, the Newton polygon for e^z *is composed of all the nodes* $(j, -\log j!)$, *for* $j \in \mathbb{Z}$. *Hence, we may apply Theorem 5.19 to compute the Newton polygon for* $f(z)$, *and then use Theorem 5.16 to construct inclusion results. The Newton polygon yields the following inclusions:*



(a) (b)

Figure 5.6: In Figure 5.6a the Newton polygon of $t_\times e^x$ (in blue) and the one associated with $t_\times g(x)$ (in red). In Figure 5.6b the exclusion annuli obtained from Theorem 5.16 and the approximated roots.

- *The open disc of radius* $r \approx 0.038$ *and centered at zero, corresponding to the slope of the first segment of the Newton polygon, does not contain any root, as predicted by Theorem 5.17.*
- *Similarly, the annulus* $\mathring{\mathcal{A}}(0.174, 4.59)$ *does not contain any root; in addition, there is exactly one root in the disc of radius* 0.174*, very close to the boundary of the disc mentioned above.*
- *Finally, the annulus* $\mathring{\mathcal{A}}(2.40, 5.27)$ *does not contain roots, and there are exactly two roots in the annulus* $\mathcal{A}(4.59, 5.27)$*.*

These inclusions are displayed in the right plot of Figure 5.6.

Example 5.8. *Let*

$$f(z) = e^z + e^{\frac{1}{z}} - 1,$$

and let $f_n(z) = \sum_{j=-n}^{n} b_j z^j$ *be a truncation of* $f(z)$ *expressed as a Laurent series. Consider the function* $g(z) = f_n(z) + p(z)$ *where* $p(z)$ *is the Laurent polynomial defined by*

$$p(z) := e^6 z^{-9} + e^{12} z^{-3} + e + e^2 z^2 + e^{-10} z^4 + e^{-14} z^5 + e^{-20} z^5.$$

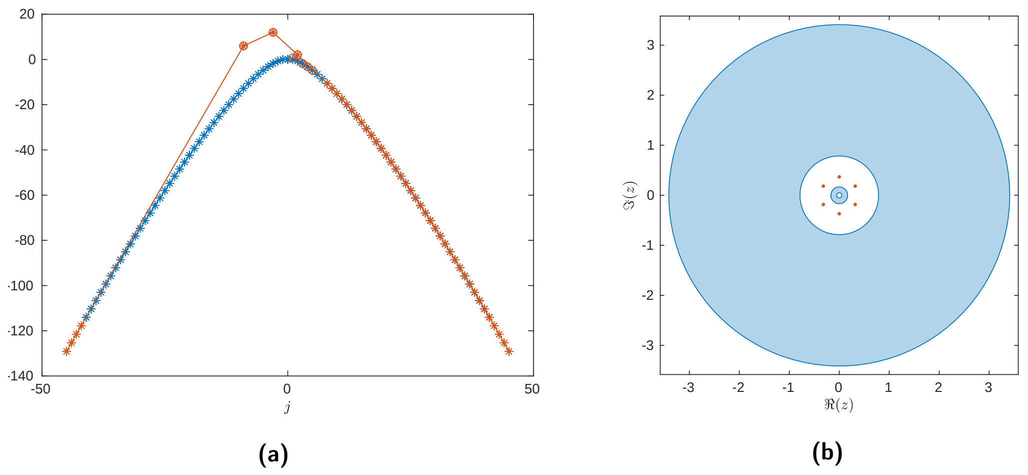


Figure 5.7: In Figure 5.7a the Newton polygon of $t_\times f_{45}(x)$ (in blue) and the one associated with $t_\times g(x)$ (in red). In Figure 5.7b the exclusion annuli obtained from Theorem 5.16 and the approximated roots.

Similarly to Example 5.7, the Newton Polygon for $t_\times f(x)$ is easily determined, and is composed by the nodes of coordinates $(j, -\log(|j|!))$ for $|j| < n$. Here, we set $n = 45$ and computed numerically the updated Newton polygon using Theorem 5.19. This yields a Newton polygon with fewer nodes, with the following inclusion/exclusion annuli:

- *The annulus $\mathcal{A}_1 := \mathring{\mathcal{A}}(0.05, 0.17)$ does not contain any root, and the disc inside it contains 34 roots minus poles.*
- *The annulus $\mathcal{A}_2 = \mathring{\mathcal{A}}(0.79, 3.41)$ does not contain any root, and exactly 6 roots are contained between $\mathcal{A}_1$ and $\mathcal{A}_2$.*

The inclusions, and the corresponding roots, are visible in Figure 5.7.

In this second set of examples we go back to the origin story of this chapter. In Section 3.4 we focused on how important the quadrature rule is for the contour integral algorithms. Nevertheless theoretical results on how to choose the number of quadrature points N are still in an early stage. In the example of Section 3.4.1 we witnessed how the relative errors between two quadrature approximations with an increasing number of points is not a good metric to predict the final backward error of the eigenpairs. Later, we saw how Van Barel and Kravanja associated a filter function $b_0(z)$ to the trapezoidal quadrature rule on Ω [Van16; VK16]. They explained that the quality of the approximation depends not only on the number of quadrature points N, but also on the distance between $\partial\Omega$ and the eigenvalues outside Ω. Furthermore, we proved that their analysis easily extends to the meromorphic case and we derived a link between the quadrature parameters in the Loewner interpretation (see Equation (3.19)). Unfortunately, at the time we did not have the tool to provide a similar result for the classical interpretation of Beyn's algorithm. We simply wrote that if the eigenvalues lie near $\partial\Omega$, then the contour algorithms need several quadrature points N, while we can set N to be quite small (e.g., $N = 4, 8$) in the opposite scenario. Theorem 5.16 is the perfect tool for this task. Under its hypotheses, the annuli of exclusion can help us set the optimal number of quadrature points N for a specific eigenvalue problem. The same idea can be applied in the polynomial eigenvalue problems by using its original version [NST15, Theorem 2.7]. The following examples will clarify the procedure.

Example 5.9. *Consider a matrix polynomial $P(z) = \sum_{j=0}^{4} B_0 z^j$ generated with the* MATLAB *commands*

```
rng(42); n = 20;
B0 = randn(n); B1 = 1e5*randn(n);
B2 = randn(n); B3 = 1e-2*randn(n); B4 = 1e3*randn(n);
```

The associated tropical polynomial $\mathsf{t}_\times P(z)$ (using the spectral norm) has two roots

$$\alpha_1 = \|B_0\|_2/\|B_1\|_2 \approx 10^{-5}, \qquad \alpha_2 = (\|B_1\|_2/\|B_4\|_2)^{1/3} \approx 4.8$$

of multiplicity 1 and 3, respectively. It holds that $\delta_1 \ll (1+2\kappa(B_0))^{-2} \approx 10^{-5}$, hence there are 20 eigenvalues in $\mathcal{D}((1+2\kappa(B_0))\alpha_0)$, and no eigenvalues in the annulus

$$\mathring{\mathcal{A}}((1+2\kappa(B_0))\alpha_0, (1+2\kappa(B_0))^{-1}\alpha_1) \approx \mathring{\mathcal{A}}(0.0012, 0.039).$$

If we are interested in the eigenvalues inside $\Omega = \mathcal{D}((1+2\kappa(B_0))\alpha_1) := \mathcal{D}(r)$, then we can set $N = 10$, given that

$$b_0(z) = \frac{1}{1-(\frac{z}{r})^{12}} \approx \mathcal{O}(10^{-16})$$

when $|z| \approx 0.039$. In fact, our contour solver (without any refinement) returns all the eigenvalues with a backward error of the order of machine precision.

Example 5.10. *Consider again the polynomial $P(z)$ of Example 5.9 and the holomorphic function $F(z) = zI + \mathrm{e}^{-z}B$, where $B \in \mathbb{R}^{20\times 20}$ is a randomly generated matrix. We want to find the tropical roots of $\mathrm{t}_\times(P(x) + F(x)) := \mathrm{t}_\times G(x)$. Note that $\mathrm{t}_\times G(x) = \sup_{j\in\mathbb{N}} \|C_j\|_2 x^j$ with*

$$\begin{cases} C_j = B_j + \dfrac{B}{j!}, & \text{for } j = 0,2,3, \\ C_1 = B_1 + B + I, \\ C_j = \dfrac{B}{j!}, & \text{for } j > 4. \end{cases}$$

In Figure 5.8 we plotted the Newton polygon for $\mathrm{t}_\times G(x)$. There we can see the first three tropical roots

$$\alpha_1 = \|C_0\|_2/\|C_1\|_2 \approx 2\cdot 10^{-5},$$
$$\alpha_2 = (\|C_1\|_2/\|C_4\|_2)^{1/3} \approx 4.8,$$
$$\alpha_3 = (\|C_4\|_2/\|C_{21}\|_2)^{1/17} \approx 21.3.$$

By following the same steps of Example 5.9, we can compute $(1+2\kappa(C_0))\alpha_1 \approx 0.004$ and $(1+2\kappa(C_0))^{-1}\alpha_2 \approx 0.025$ and we infer there are 20 eigenvalues in $\mathcal{D}(0.004)$ and no eigenvalues in $\mathring{\mathcal{A}}(0.004, 0.025)$. Hence we can set $N = 18$ so that $b_0(z) = \mathcal{O}(10^{-15})$ for $|z| \approx 0.025$. Once again, our contour solver returns all the 20 eigenvalues to machine precision without any refinement.

5.5 FINAL REMARKS

In this chapter we generalised the concept of tropical roots to Laurent series $\mathrm{t}_\times f(x)$. We showed that, in order to have a consistent theoretical system, the tropical roots of $\mathrm{t}_\times f(x)$ consist not only of the points of nondifferentiability, but also of the extrema of

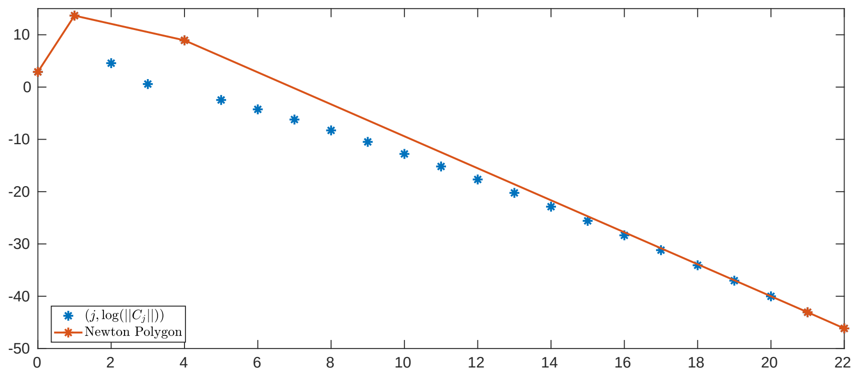


Figure 5.8: The (truncated) Newton polygon of $t_\times G(x)$. We can see the first 3 tropical roots.

the interval where $t_\times f(x)$ is a real-valued function. Later we expanded the localisation results stated for scalar polynomials in [Sha11] and for matrix-valued polynomials in [NST15]. In addition, we showed how to update the Newton polygon of a Laurent function $t_\times f(x)$ to obtain the Newton polygon of $t_\times(f(x) + p(x))$, where $p(z)$ is a polynomial, and we proved that the strategy terminates almost surely. Finally, we linked these localisation results to contour algorithm solvers and we described how one can use the annuli of exclusions to select the number of quadrature points in Beyn's algorithm and its interpretation under the right hypotheses.

6 CONCLUSIONS

The main goal of this thesis was the development of a general solver for holomorphic and meromorphic eigenvalue problems of the form $F(\lambda)v = 0$ for all the λ in a target region Ω. Previous works on this subject divided the solvers in three categories: the ones based on the Newton method, the ones based on rational approximations, and the ones based on contour integral solvers. Here we focused on the latter two classes, since the Newton method does not allow natively to return all the eigenvalues in Ω.

In Chapter 2 we analysed the generalisation of contour integral solvers to meromorphic eigenvalue problems, which up to now were only used in the holomorphic case. We proved that under "real-life" hypotheses, an everyday user would not distinguish whether the underlying function $F(z)$ is holomorphic or meromorphic; nevertheless, we also showed that in the general case a contour solver may return some of the poles of $F(z)$ as its eigenvalues.

Chapter 3 is the ideal continuation of Chapter 2. After having generalised the theoretical results to meromorphic functions, we aimed to develop an algorithm for mid-sized nonlinear eigenvalue problems. In fact, nowadays many algorithms for large-scale problems plan to project them on smaller ones, and then tackle them with another solver. Nevertheless, such an algorithm for a general nonlinear eigenvalue problem was still missing, therefore we put particular emphasis on the automatic choice of the parameters, and on its ability to return eigenvalues with the desired backward error. When the contour solver is not able to fulfil its duty, a couple of steps of the Newton method are used to refine the results. In the future we should focus on interweaving the algorithm with the theory of Chapter 5 and on implementing it in a faster way, exploiting its parallelisable nature. In addition, other authors have been working on similar algorithms: among them, we recall Krenner and Polizzi who developed a hybrid algorithm between NLFEAST and Beyn's approach [BP20].

Therefore future works should focus on understanding which path is the best to follow.

In Chapter 4 we contributed to the algorithms based on rational approximations. These eigensolvers require two main steps: first, one must approximate in some sense the original function $F(z)$ with a rational function $R(z)$; then, one must solve the approximate eigenproblem $R(\lambda)v = 0$. There, we focused on the former step, with the idea that the quality of these eigensolvers depends primarily on the initial approximation. We developed a precise error analysis for the backward error of the approximated eigenpairs and with this in mind we proposed two robust approximation algorithms, one for functions expressed in split form, the other for functions expressed in black-box form. The former, named weighted AAA, is an enhancement of the set-valued AAA algorithm, which considers the norms of the matrix coefficients to return an approximant with the same precision, but lower degree. The latter is a two-step algorithm which uses the surrogate AAA algorithm in the first phase and then refines the approximant with a Leja–Bagby sampling. In this way we combine together the ability of AAA of not using prescribed poles and the robust convergence of the Leja–Bagby sampling. Future works on this subject should concentrate on the solution of the approximated eigenproblem. For this thesis we have leaned on either the commands `eig` or `eigs` from MATLAB, and thus a more careful implementation of the linear eigensolver should return better results.

Finally, we dedicated Chapter 5 to tropical algebra. The origin of this chapter lay in the problem of finding a suitable number of quadrature points for the contour solvers of Chapter 3. We thought that the annuli of exclusion derived from tropical roots were good candidates to solve this issue under suitable hypotheses. On one hand, up to this moment the theory was developed only for polynomials; on the other hand, after a careful analysis of [NST15], it seemed really plausible that the same results would hold in the meromorphic case. As it usually happens in these situations, this was almost true: the generalisation was not exactly straightforward and it needed many details to be fixed. Nevertheless, there is always a bright side: we understood the deep relationship between tropical roots and the radii of convergence of Laurent series and we discovered more results than initially planned. Future works can focus on applications where tropical roots of polynomials are already in use, but the generalisation to meromorphic functions would be useful.

This marks the end of the thesis and it seems a nice place to stop. On a more personal side, writing this thesis up was the perfect closure of my PhD in Manchester: dear reader, if you have arrived at this point, I hope that going through this work brought you at least half the pleasure that completing it brought to me.

BIBLIOGRAPHY

[ACL09] A. Amiraslani, R. M. Corless, and P. Lancaster. "Linearization of matrix polynomials expressed in polynomial bases." In: *IMA J. Numer. Anal.* 29.1 (2009), pp. 141–157.

[ACL95] A. L. Andrew, E. K. Chu, and P. Lancaster. "On the numerical solution of nonlinear eigenvalue problems." In: *Computing* 55.2 (1995), pp. 91–111.

[AF21] M. J. Ablowitz and A. S. Fokas. *Introduction to complex variables and applications*. Cambridge Texts in Applied Mathematics. Cambridge University Press, Cambridge, 2021, pp. viii+411.

[AGL08] M. Akian, S. Gaubert, and A. Lakhoua. "The max-plus finite element method for solving deterministic optimal control problems: basic properties and convergence analysis." In: *SIAM Journal on Control and Optimization* 47.2 (2008), pp. 817–848.

[Ant11] P. Antunes. "On the buckling eigenvalue problem." In: *Journal of Physics A: Mathematical and Theoretical* 44 (Apr. 2011), p. 215205.

[AS01] A. C. Antoulas and D. C. Sorensen. "Approximation of large-scale dynamical systems: an overview." In: vol. 11. 5. Numerical analysis and systems theory (Perpignan, 2000). 2001, pp. 1093–1121.

[AS64] M. Abramowitz and I. A. Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. Vol. 55. National Bureau of Standards Applied Mathematics Series. For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C., 1964, pp. xiv+1046.

[Asa+09] J. Asakura et al. "A numerical method for nonlinear eigenvalue problems using contour integrals." In: *JSIAM Lett.* 1 (2009), pp. 52–55.

[Asa+10] J. Asakura et al. “A numerical method for polynomial eigenvalue problems using contour integral.” In: *Jpn. J. Ind. Appl. Math.* 27.1 (2010), pp. 73–90.

[AT11] H. Avron and S. Toledo. “Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix.” In: *J. ACM* 58.2 (2011), Art. 8, 17.

[AV06] D. Arthur and S. Vassilvitskii. “How Slow is the k-Means Method?” In: *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry*. SCG ’06. Sedona, Arizona, USA: Association for Computing Machinery, 2006, pp. 144–153. ISBN: 1595933409.

[Bac+92] F. L. Baccelli et al. *Synchronization and linearity: an algebra for discrete event systems*. Wiley Series in Probability and Mathematical Statistics: Probability and Mathematical Statistics. An algebra for discrete event systems. John Wiley & Sons, Ltd., Chichester, 1992, pp. xx+489.

[Bag67] T. Bagby. “The Modulus of a Plane Condenser.” In: *Journal of Mathematics and Mechanics* 17.4 (1967), pp. 315–329.

[Bag69] T. Bagby. “On interpolation by rational functions.” In: *Duke Math. J.* 36 (1969), pp. 95–104.

[BEG20] M. C. Brennan, M. Embree, and S. Gugercin. “Contour Integral Methods for Nonlinear Eigenvalue Problems: A Systems Theoretic Approach.” In: (2020). arXiv: `2012.14979 [math.NA]`.

[BEK11] W.-J. Beyn, C. Effenberger, and D. Kressner. “Continuation of eigenvalues and invariant pairs for parameterized nonlinear eigenvalue problems.” In: *Numer. Math.* 119.3 (2011), pp. 489–516.

[Bet+11] T. Betcke et al. *NLEVP: A Collection of Nonlinear Eigenvalue Problems*. MIMS EPrint 2011.116. UK: Manchester Institute for Mathematical Sciences, The University of Manchester, Dec. 2011, p. 27.

[Bet08] T. Betcke. “Optimal Scaling of Generalized and Polynomial Eigenvalue Problems.” In: *SIAM J. Matrix Analysis Applications* 30 (Jan. 2008), pp. 1320–1338.

[Bey12] W.-J. Beyn. “An integral method for solving nonlinear eigenvalue problems.” In: *Linear Algebra Appl.* 436.10 (2012), pp. 3839–3863.

[Bin96] D. A. Bini. “Numerical computation of polynomial zeros by means of Aberth’s method.” In: *Numerical Algorithms* 13 (Feb. 1996), pp. 179–200.

[BN13] D. A. Bini and V. Noferini. “Solving polynomial eigenvalue problems by means of the Ehrlich–Aberth method.” In: *Linear Algebra and its Applications* 439.4 (2013). 17th Conference of the International Linear Algebra Society, Braunschweig, Germany, August 2011, pp. 1130–1149.

[BNS13] D. A. Bini, V. Noferini, and M. Sharify. “Locating the eigenvalues of matrix polynomials.” In: *SIAM Journal on Matrix Analysis and Applications* 34.4 (2013), pp. 1708–1727.

[BP20] J. Brenneck and E. Polizzi. *An Iterative Method for Contour-Based Nonlinear Eigensolvers*. 2020. arXiv: `2007.03000 [math.NA]`.

[Bra06] M. Brand. “Fast low-rank modifications of the thin singular value decomposition.” In: *Linear Algebra Appl.* 415.1 (2006), pp. 20–30.

[Bre18] M. C. Brennan. “Rational Interpolation Methods for Nonlinear Eigenvalue Problems.” PhD thesis. Virginia Tech, 2018.

[But10] P. Butkovič. *Max-linear systems: theory and algorithms*. Springer Science & Business Media, 2010.

[BW73] K.-J. Bathe and E. L. Wilson. “Solution methods for eigenvalue problems in structural mechanics.” In: *International Journal for Numerical Methods in Engineering* 6.2 (1973), pp. 213–226.

[Car95] H. Cartan. *Elementary theory of analytic functions of one or several complex variables*. Translated from the French, Reprint of the 1973 edition. Dover Publications, Inc., New York, 1995, p. 228.

[Cha+19] E. Y. S. Chan et al. "Algebraic linearizations of matrix polynomials." In: *Linear Algebra Appl.* 563 (2019), pp. 373–399.

[Che+17] H. Chen et al. "Improving the numerical stability of the Sakurai-Sugiura method for quadratic eigenvalue problems." In: *JSIAM Lett.* 9 (2017), pp. 17–20.

[Che98] E. W. Cheney. *Introduction to approximation theory*. Reprint of the second (1982) edition. AMS Chelsea Publishing, Providence, RI, 1998, pp. xii+259. ISBN: 0-8218-1374-9.

[CK20] A. Cortinovis and D. Kressner. *On randomized trace estimates for indefinite matrices with an application to determinants*. 2020. arXiv: 2005.10009 [math.NA].

[CM80] R. A. Cuninghame-Green and P. F. J. Meijer. "An algebra for piecewise-linear minimax problems." In: *Discrete Appl. Math.* 2.4 (1980), pp. 267–294.

[CR19] C. Campos and J. E. Roman. *NEP: a module for the parallel solution of nonlinear eigenvalue problems in SLEPc*. 2019.

[Dav97] T. A. Davis. "The University of Florida sparse matrix collection." In: *NA DIGEST* (1997).

[DH11] T. A. Davis and Y. Hu. "The University of Florida Sparse Matrix Collection." In: *ACM Trans. Math. Softw.* 38.1 (Dec. 2011).

[DPS16] E. Di Napoli, E. Polizzi, and Y. Saad. "Efficient estimation of eigenvalue counts in an interval." In: *Numerical Linear Algebra with Applications* 23.4 (2016), pp. 674–692.

[DR07] P. J. Davis and P. Rabinowitz. *Methods of numerical integration*. Corrected reprint of the second (1984) edition. Dover Publications, Inc., Mineola, NY, 2007, pp. xii+612.

[Eff13] C. Effenberger. "Robust successive computation of eigenpairs for nonlinear eigenvalue problems." In: *SIAM J. Matrix Anal. Appl.* 34.3 (2013), pp. 1231–1256.

[EG19] S. Elsworth and S. Güttel. “Conversions between barycentric, RKFUN, and Newton representations of rational interpolants.” In: *Linear Algebra Appl.* 576 (2019), pp. 246–257.

[EMS20] M. El-Guide, A. Miedlar, and Y. Saad. “A rational approximation method for solving acoustic nonlinear eigenvalue problems.” In: *Engineering Analysis with Boundary Elements* 111 (Feb. 2020), pp. 44–54.

[Fra61a] J. G. F. Francis. “The *QR* transformation: a unitary analogue to the *LR* transformation. I.” In: *Comput. J.* 4 (1961), pp. 265–271.

[Fra61b] J. G. F. Francis. “The *QR* transformation. II.” In: *Comput. J.* 4 (1961), pp. 332–345.

[Fro69] M. Froissart. “Approximation de Padé: application à la physique des particules élémentaires.” In: *RCP, Programme No. 25*. Vol. 9. CNRS, Strasbourg, 1969, pp. 1–13.

[Fro79] F. G. Frobenius. “Theorie der linearen Formen mit ganzen Coefficienten.” In: *J. Reine Angew. Math.* 86 (1879), pp. 146–208.

[GJW82] M. R. Garey, D. S. Johnson, and H. Witsenhausen. “The complexity of the generalized Lloyd–Max problem.” In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 255–256.

[GMP18] B. Gavin, A. Międlar, and E. Polizzi. “FEAST eigensolver for nonlinear eigenvalue problems.” In: *J. Comput. Sci.* 27 (2018), pp. 107–117.

[GR04] H. Guo and R. A. Renaut. “Estimation of $\mathbf{u}^T f(A)\mathbf{v}$ for large-scale unsymmetric matrices.” In: *Numer. Linear Algebra Appl.* 11.1 (2004), pp. 75–89.

[Gra72] R. L. Graham. “An efficient algorithm for determining the convex hull of a finite planar set.” In: *Info. Pro. Lett.* 1 (1972), pp. 132–133.

[GS09] S. Gaubert and M. Sharify. “Tropical scaling of polynomial matrices.” In: *Positive systems*. Vol. 389. Lect. Notes Control Inf. Sci. Springer, Berlin, 2009, pp. 291–303.

[GS71] I. C. Gohberg and E. I. Sigal. "An operator generalization of the logarithmic residue theorem and Rouché's theorem." In: *Mat. Sb. (N.S.)* 84(126) (1971), pp. 607–629.

[GT17] S. Güttel and F. Tisseur. "The nonlinear eigenvalue problem." In: *Acta Numer.* 26 (2017), pp. 1–94.

[Güt+14] S. Güttel et al. "NLEIGS: A Class of Fully Rational Krylov Methods for Nonlinear Eigenvalue Problems." In: *SIAM Journal on Scientific Computing* 36.6 (2014), A2842–A2864.

[Güt13] S. Güttel. "Rational Krylov approximation of matrix functions: Numerical methods and optimal pole selection." In: *GAMM Mitteilungen* 36 (Aug. 2013).

[GV96] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Third. The Johns Hopkins University Press, 1996.

[GW06] K. Green and T. Wagenknecht. "Pseudospectra and delay differential equations." In: *J. Comput. Appl. Math.* 196.2 (2006), pp. 567–578.

[Had67] K. P. Hadeler. "Mehrparametrige und nichtlineare Eigenwertaufgaben." In: *Arch. Rational Mech. Anal.* 27 (1967), pp. 306–328.

[HBS10] X. Huang, Z. Bai, and Y. Su. "Nonlinear rank-one modification of the symmetric eigenvalue problem." In: *J. Comput. Math.* 28.2 (2010), pp. 218–234.

[Hig08] N. J. Higham. *Functions of matrices*. Theory and computation. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2008, pp. xx+425. ISBN: 978-0-89871-646-7.

[Hig90] N. J. Higham. "Analysis of the Cholesky decomposition of a semi-definite matrix." In: *in Reliable Numerical Computation*. Oxford University Press, 1990, pp. 161–185.

[HL08] S. Hammarling and C. Lucas. "Updating the QR factorization and the least squares problem." In: *MIMS Eprint* (2008).

[HMT13] S. Hammarling, C. J. Munro, and F. Tisseur. "An algorithm for the complete solution of quadratic eigenvalue problems." In: *ACM Trans. Math. Software* 39.3 (2013), Art. 18, 19.

[HNT19] N. J. Higham, G. M. Negri Porzio, and F. Tisseur. *An Updated Set of Nonlinear Eigenvalue Problem*. MIMS EPrint 2019.05. UK: Manchester Institute for Mathematical Sciences, The University of Manchester, Mar. 2019, p. 26.

[Hoc17] A. Hochman. "FastAAA: A fast rational-function fitter." In: *2017 IEEE 26th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*. 2017, pp. 1–3.

[HSY17] R. Huang, J. Sun, and C. Yang. "Recursive Integral Method with Cayley Transformation." In: *Numerical Linear Algebra with Applications* (May 2017).

[HT03] N. J. Higham and F. Tisseur. "Bounds for eigenvalues of matrix polynomials." In: *Linear algebra and its applications* 358.1-3 (2003), pp. 5–22.

[Hua+16] R. Huang et al. "Recursive integral method for transmission eigenvalues." In: *Journal of Computational Physics* 327 (2016), pp. 830–840.

[Hua19] R. Huang. "Novel Computation Methods for Eigenvalue Problems." PhD thesis. Michigan Technological University, 2019.

[Hut90] M. F. Hutchinson. "A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines." In: *Comm. Statist. Simulation Comput.* 19.2 (1990), pp. 433–450.

[Ips97] I. C. F. Ipsen. "Computing an eigenvector with inverse iteration." In: *SIAM Rev.* 39.2 (1997), pp. 254–291.

[Jar+18] E. Jarlebring et al. *NEP-PACK: A Julia package for nonlinear eigenproblems.* Available at https://github.com/nep-pack. 2018.

[Jar12] E. Jarlebring. "Convergence factors of Newton methods for nonlinear eigenvalue problems." In: *Linear Algebra Appl.* 436.10 (2012), pp. 3943–3953.

[JM10] E. Jarlebring and W. Michiels. "Invariance properties in the root sensitivity of time-delay systems with double imaginary roots." In: *Automatica J. IFAC* 46.6 (2010), pp. 1112–1115.

[Kar00] S. V. Kartalopoulos. *Introduction to DWDM technology: data in a rainbow*. SPIE Optical Engineering Press, 2000.

[Kel71] M. V. Keldysh. "The completeness of eigenfunctions of certain classes of nonselfadjoint linear operators." In: *Uspehi Mat. Nauk* 26.4(160) (1971), pp. 15–41.

[KM99] V. Kozlov and V. Maz'ja. *Differential equations with operator coefficients with applications to boundary value problems for partial differential equations*. Springer Monographs in Mathematics. Springer-Verlag, Berlin, 1999.

[Kub61] V. N. Kublanovskaja. "Certain algorithms for the solution of the complete problem of eigenvalues." In: *Soviet Math. Dokl.* 2 (1961), pp. 17–19.

[Kub70] V. N. Kublanovskaya. "On an approach to the solution of the generalized latent value problem for λ-matrices." In: *SIAM J. Numer. Anal.* 7 (1970), pp. 532–537.

[Lan02] P. Lancaster. *Lambda-matrices and vibrating systems*. Reprint of the 1966 original [Pergamon Press, New York; MR0210345 (35 #1238)]. Dover Publications, Inc., Mineola, NY, 2002, pp. xx+193. ISBN: 0-486-42546-0.

[LC56] R. K. Livesley and D. B. Chandler. "Stability Functions for Structural Frameworks." In: *Manchester University Press* (1956).

[Lie+18] P. Lietaert et al. "Automatic rational approximation and linearization of nonlinear eigenvalue problems." In: *arXiv e-prints*, arXiv:1801.08622 (Jan. 2018), arXiv:1801.08622.

[LK90] C. S. Lent and D. J. Kirkner. "The quantum transmitting boundary method." In: *Journal of Applied Physics* 67.10 (1990), pp. 6353–6359.

[LS06] E. Levin and E. B. Saff. "Potential theoretic tools in polynomial and rational approximation." In: *Harmonic analysis and rational approximation*. Vol. 327. Lect. Notes Control Inf. Sci. Springer, Berlin, 2006, pp. 71–94.

[LS94] E. Levin and E. B. Saff. “Optimal ray sequences of rational functions connected with the Zolotarev problem.” In: *Constr. Approx.* 10.2 (1994), pp. 235–273.

[Mac+06a] D. S. Mackey et al. “Structured polynomial eigenvalue problems: good vibrations from good linearizations.” In: *SIAM J. Matrix Anal. Appl.* 28.4 (2006), pp. 1029–1051.

[Mac+06b] D. S. Mackey et al. “Vector spaces of linearizations for matrix polynomials.” In: *SIAM J. Matrix Anal. Appl.* 28.4 (2006), pp. 971–1004.

[McE06] W. M. McEneaney. *Max-plus methods for nonlinear control and estimation*. Springer Science & Business Media, 2006.

[Mel13] A. Melman. “Generalization and variations of Pellet’s theorem for matrix polynomials.” In: *Linear Algebra and its Applications* 439.5 (2013), pp. 1550–1567.

[MFS11] Y. Maeda, Y. Futamura, and T. Sakurai. “Stochastic estimation method of eigenvalue density for nonlinear eigenvalue problem on the complex plane.” In: *JSIAM Lett.* 3 (2011), pp. 61–64.

[MM03] R. Mennicken and M. Möller. *Non-self-adjoint boundary eigenvalue problems*. Vol. 192. North-Holland Mathematics Studies. North-Holland Publishing Co., Amsterdam, 2003, pp. xviii+500. ISBN: 0-444-51447-3.

[MN07] W. Michiels and S.-I. Niculescu. *Stability and stabilization of time-delay systems*. Vol. 12. Advances in Design and Control. An eigenvalue-based approach. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2007, pp. xxii+378. ISBN: 978-0-898716-32-0.

[MNV09] M. Mahajan, P. Nimbhorkar, and K. Varadarajan. “The Planar k-Means Problem is NP-Hard.” In: *WALCOM: Algorithms and Computation*. Ed. by S. Das and R. Uehara. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 274–285. ISBN: 978-3-642-00202-1.

[Mor20] K. Morikuni. *Projection method for interior eigenproblems of linear nonsquare matrix pencils*. 2020.

[MS73] C. B. Moler and G. W. Stewart. "An Algorithm for Generalized Matrix Eigenvalue Problems." In: *SIAM Journal on Numerical Analysis* 10.2 (1973), pp. 241–256.

[NST15] V. Noferini, M. Sharify, and F. Tisseur. "Tropical roots as approximations to eigenvalues of matrix polynomials." In: *SIAM J. Matrix Anal. Appl.* 36.1 (2015), pp. 138–157.

[NST18] Y. Nakatsukasa, O. Sète, and L. N. Trefethen. "The AAA algorithm for rational approximation." In: *SIAM J. Sci. Comput.* 40.3 (2018), A1494–A1522.

[Ors71] S. A. Orszag. "Accurate solution of the Orr–Sommerfeld Stability Equation." In: *J. Fluid Mech.* 50.4 (1971), pp. 689–703.

[Pol09] E. Polizzi. "Density-matrix-based algorithm for solving eigenvalue problems." In: *Phys. Rev. B* 79 (11 Mar. 2009), p. 115112.

[RA15] F. Roosta-Khorasani and U. Ascher. "Improved bounds on sample size for implicit matrix trace estimators." In: *Found. Comput. Math.* 15.5 (2015), pp. 1187–1212.

[Roc70] R. T. Rockafellar. *Convex analysis*. Princeton Mathematical Series, No. 28. Princeton University Press, Princeton, N.J., 1970, pp. xviii+451.

[RST05] J. Richter-Gebert, B. Sturmfels, and T. Theobald. "First steps in tropical geometry." In: *Contemporary Mathematics* 377 (2005), pp. 289–318.

[Ruh73] A. Ruhe. "Algorithms for the nonlinear eigenvalue problem." In: *SIAM J. Numer. Anal.* 10 (1973), pp. 674–689.

[SB11] Y. Su and Z. Bai. "Solving rational eigenvalue problems via linearization." In: *SIAM J. Matrix Anal. Appl.* 32.1 (2011), pp. 201–216.

[SFT13] T. Sakurai, Y. Futamura, and H. Tadano. "Efficient parameter estimation and implementation of a contour integral-based eigensolver." In: *Journal of Algorithms & Computational Technology* 7.3 (2013), pp. 249–269.

[Sha11] M. Sharify. "Scaling Algorithms and Tropical Methods in Numerical Matrix Analysis." PhD thesis. École Polytechnique, 2011.

[Sim78] I. Simon. “Limited subsets of a free monoid.” In: *19th Annual Symposium on Foundations of Computer Science (sfcs 1978)*. 1978, pp. 143–150.

[SS03] T. Sakurai and H. Sugiura. “A projection method for generalized eigenvalue problems using numerical integration.” In: *Proceedings of the 6th Japan-China Joint Seminar on Numerical Mathematics (Tsukuba, 2002)*. Vol. 159. 1. 2003, pp. 119–128.

[SS61] H. J. S. Smith and J. J. Sylvester. “XV. On systems of linear indeterminate equations and congruences.” In: *Philosophical Transactions of the Royal Society of London* 151 (1861), pp. 293–326.

[TKL05] M. Z. Tokar, F. A. Kelly, and X. Loozen. “Role of Thermal Instabilities and Anomalous Transport in Threshold of Detachment and Mulitfacetted Asymmetric Radiation from the edge (MARFE).” In: *Physics of Plasmas* 12.052510 (2005).

[TM01] F. Tisseur and K. Meerbergen. “The quadratic eigenvalue problem.” In: *SIAM Rev.* 43.2 (2001), pp. 235–286.

[Tro68] V. P. Trofimov. “The root subspaces of operators that depend analytically on a parameter.” In: *Mat. Issled.* 3.vyp. 3 (9) (1968), pp. 117–125.

[TV21] F. Tisseur and M. Van Barel. “Min-max elementwise backward error for roots of polynomials and a corresponding backward stable root finder.” In: *Linear Algebra Appl.* 623 (2021), pp. 454–477.

[TW14] L. N. Trefethen and J. A. C. Weideman. “The exponentially convergent trapezoidal rule.” In: *SIAM Rev.* 56.3 (2014), pp. 385–458.

[UCS17] S. Ubaru, J. Chen, and Y. Saad. “Fast estimation of $\mathtt{tr}(f(A))$ via stochastic Lanczos quadrature.” In: *SIAM J. Matrix Anal. Appl.* 38.4 (2017), pp. 1075–1099.

[Ung50] H. Unger. “Nichtlineare Behandlung von Eigenwertaufgaben.” In: *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 30.8-9 (1950), pp. 281–282.

[Van+14] W. G. Vandenberghe et al. “Determining bound states in a semiconductor device with contacts using a nonlinear eigenvalue solver.” In: *Journal of Computational Electronics* 13.3 (2014), pp. 753–762.

[Van16] M. Van Barel. “Designing rational filter functions for solving eigenvalue problems by contour integration.” In: *Linear Algebra Appl.* 502 (2016), pp. 346–365.

[VK16] M. Van Barel and P. Kravanja. “Nonlinear eigenvalue problems and contour integrals.” In: *J. Comput. Appl. Math.* 292 (2016), pp. 526–540.

[VT18] M. Van Barel and F. Tisseur. “Polynomial eigenvalue solver based on tropically scaled Lagrange linearization.” In: *Linear Algebra Appl.* 542 (2018), pp. 186–208.

[Wal35] J. L. Walsh. *Interpolation and approximation by rational functions in the complex domain; 3rd ed.* Colloquium publications. Providence, RI: American Mathematical Society, 1935.

[Wil68] J. H. Wilkinson. “A Priori Error Analysis of Algebraic Processes.” In: *Proceedings International Congress Math. (Moscow: Izdat. Mir).* 1968, pp. 629–639.

[WW71] W. H. Wittrick and F. W. Williams. “A general algorithm for computing natural frequencies of elastic structures.” In: *Quart. J. Mech. Appl. Math.* 24 (1971), pp. 263–284.

[WW73] W. H. Wittrick and F. W. Williams. “An algorithm for computing critical buckling loads of elastic structures.” In: *Journal of Structural Mechanics* 1.4 (1973), pp. 497–518.

[WWZ14] K. Wimmer, Y. Wu, and P. Zhang. “Optimal query complexity for estimating the trace of a matrix.” In: *Automata, languages, and programming. Part I.* Vol. 8572. Lecture Notes in Comput. Sci. Springer, Heidelberg, 2014, pp. 1051–1062.

[YS13] S. Yokota and T. Sakurai. “A projection method for nonlinear eigenvalue problems using contour integrals.” In: *JSIAM Lett.* 5 (2013), pp. 41–44.

[ZL08] B. Zhang and Y. Li. “A Method for Calibrating the Central Catadioptric Camera via Homographic Matrix.” In: *Proceedings of the 2008 IEEE International Conference on Information and Automation, Zhangjiajie, China*. 2008, pp. 972–977.

[ZS14] L. Zeng and Y. Su. “A backward stable algorithm for quadratic eigenvalue problems.” In: *SIAM J. Matrix Anal. Appl.* 35.2 (2014), pp. 499–516.