Rational Interpolation Methods for Nonlinear Eigenvalue Problems

Michael Brennan

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Mathematics

Mark P. Embree, Co-Chair

Serkan Gugercin, Co-Chair

Christopher A. Beattie

August 14, 2018

Blacksburg, Virginia

Keywords: Nonlinear Eigenvalue Problems, Contour Integration Methods, Iterative Methods, Dynamical Systems

Rational Interpolation Methods for Nonlinear Eigenvalue Problems

Michael Brennan

(ABSTRACT)

This thesis investigates the numerical treatment of nonlinear eigenvalue problems. These problems are defined by the condition $\mathbf{T}(\lambda)\mathbf{v} = \mathbf{0}$, with $\mathbf{T} : \mathbb{C} \to \mathbb{C}^{n\times n}$, where we seek to compute the scalar-vector pairs, $\lambda \in \mathbb{C}$ and nonzero $\mathbf{v} \in \mathbb{C}^n$. The first contribution of this work connects recent contour integration methods to the theory and practice of system identification. This observation leads us to explore rational interpolation for system realization, producing a Loewner matrix contour integration technique. The second development of this work studies the application of rational interpolation to the function $\mathbf{T}(z)^{-1}$, where we use the poles of this interpolant to approximate the eigenvalues of $\mathbf{T}$. We then expand this idea to several iterative methods, where at each step the approximate eigenvalues are taken to be new interpolation points. We show that the case where one interpolation point is used is theoretically equivalent to Newton's method for a particular scalar function.

Rational Interpolation Methods for Nonlinear Eigenvalue Problems

Michael Brennan

(GENERAL AUDIENCE ABSTRACT)

This thesis investigates the numerical treatment of nonlinear eigenvalue problems. The solutions to these problems often reveal characteristics of an underlying physical system. One popular methodology for handling these problems uses contour integrals to compute a set of the solutions. The first contribution of this work connects these contour integration methods to the theory and practice of system identification. This leads us to explore other techniques for system identification, resulting in a new method.

Another common methodology approximates the nonlinear problem directly. The second development of this work studies the application of rational interpolation for this purpose. We then use this idea to form several iterative methods, where at each step the approximate solutions are taken to be new interpolation points. We show that the case where one interpolation point is used is theoretically equivalent to Newton's method for a particular scalar function.

To my family, thank you for always supporting me.

Acknowledgments

To my advisor Serkan Gugercin, thank you for the lessons in mathematics research and in life. I feel lucky to have had you as a mentor through my undergraduate and Master's studies, and will always be grateful for your advice.

I also would like to thank Mark Embree for the insights and guidance he has given me, especially in regard to the presentation of this work.

I thank my fellow graduate students for distracting me from time to time. It is amazing how a 15 minute break discussing sour dough bread (and other important things) can make bugs in code more findable. Best wishes to all of you!

Finally I thank my wonderful girlfriend, Emily, for the emotional support she has given me over the course of my Master's studies.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

Mathematical modeling enables us two significant capabilities: to predict the behavior of complex physical systems and to optimize the performance of systems that we construct.

Several examples highlight the importance of prediction and simulation to human safety. Emergency responses to natural disasters depend on the prediction of a storm's strength or aftershock locations when handling severe weather patterns or seismic activity. Structural design decisions for buildings and vehicles depend on our modeling of material interactions, stresses, and deformations.

In addition, optimization techniques that minimize material waste and control laws that govern automated processes rely on accurate and efficient models that often must be evaluated in real time. Optimal solutions help reduce project costs and build structures that perform better. For these reasons, the scientific and industrial communities have invested

much effort toward the development of mathematical modeling.

The process of mathematical modeling can be split into two parts. First, one develops a mathematical description of the laws governing the phenomena of interest, known as a model. This thesis regards the next step of the process, the implementation of the model numerically, where we obtain the results needed for the application purpose. Specifically, we look to improve the treatment of *nonlinear eigenvalue problems* that are generated by dynamical systems, a common framework used to cast a mathematical model. These problems come in the form

$$\mathbf{T}(\lambda)\mathbf{v} = \mathbf{0},$$

where $\mathbf{T} : \mathbb{C} \to \mathbb{C}^{n\times n}$ denotes an analytic matrix-valued function, and one seeks to find the scalar-vector pairs, $\lambda \in \mathbb{C}$ and nonzero $\mathbf{v} \in \mathbb{C}^n$.

Many physical phenomena can be modeled in the setting of dynamical systems, coming in the form of a general degree d differential equation,

$$\mathbf{f}(\mathbf{x}^{(d)}(t), \ldots, \mathbf{x}'(t), \mathbf{x}(t), t, \mathbf{p}) = 0,$$

subject to some initial condition, where the function $\mathbf{f} : \mathbb{C}^n \times \cdots \times \mathbb{C}^n \times \mathbb{C}^n \times [0, \infty) \times \mathbb{C}^P \to \mathbb{C}^n$. One often looks to compute the state vector, $\mathbf{x}(t) \in \mathbb{C}^n$, as a function of time, where the dynamics of the model describe $\mathbf{x}(t)$ through the differential equation defined above. The dimension of the state variable, n, is referred to as the order of the dynamical system. Realistic applications commonly yield n of 10^6 or greater.

For optimization, control and uncertainty quantification, one is challenged to evaluate

this model for many choices of system parameters, represented as $\mathbf{p} \in \mathbb{C}^P$. For example, the damping coefficients in a spring-damper system may be thought of as tunable parameters, where we wish to dampen vibrations of certain frequencies optimally.

We now describe how several common dynamical systems generate eigenvalue problems, and how the solutions to these eigenvalue problems expose possible behaviors of the underlying dynamical system.

1.1 First Degree Linear Dynamical Systems and Standard Eigenvalue Problems

Consider the first degree linear differential equation

$$\mathbf{x}'(t) = \mathbf{A}\mathbf{x}(t), \tag{1.1}$$

where $\mathbf{A} \in \mathbb{C}^{n\times n}$. A common method of building analytic solutions to this problem starts by making the ansatz of

$$\mathbf{x}(t) = e^{\lambda t}\mathbf{v},$$

where $\lambda \in \mathbb{C}$ and $\mathbf{v} \in \mathbb{C}^n$ are constant. By substituting this ansatz into (1.1), one arrives at the standard eigenvalue problem:

$$(e^{\lambda t}\mathbf{v})' = \mathbf{A}(e^{\lambda t}\mathbf{v}) \implies \lambda e^{\lambda t}\mathbf{v} = e^{\lambda t}\mathbf{A}\mathbf{v} \implies \lambda\mathbf{v} = \mathbf{A}\mathbf{v}.$$

When this relationship is satisfied with $\mathbf{v} \neq \mathbf{0}$, the scalar/vector pair $(\lambda, \mathbf{v})$ is referred to as an eigenpair of $\mathbf{A}$, where λ is called an eigenvalue of $\mathbf{A}$ with corresponding eigenvector $\mathbf{v}$.

The collection of all eigenvalues is referred to as the spectrum of $\mathbf{A}$ and will be denoted by $\sigma(\mathbf{A})$.

The eigenvalue equation can be equivalently written as

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

and so we see that eigenvalues are the values that render the matrix valued function $\mathbf{A} - \lambda\mathbf{I}$ singular. Thus for λ to be an eigenvalue, we require

$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0.$$

The expression $\det(\mathbf{A} - \lambda\mathbf{I})$ is known as the characteristic polynomial of $\mathbf{A}$, i.e., eigenvalues are roots of the characteristic polynomial. From this condition we see there are n eigenvalues, accounting for repeated roots. The number of times an eigenvalue repeats is referred to as its algebraic multiplicity.

Given that (1.1) is linear, any linear combination of solutions will also be a solution. If each eigenvalue has the same number of linearly independent eigenvectors as its algebraic multiplicity, one can provide the general solution to (1.1) as

$$\mathbf{x}(t) = \sum_{j=1}^{n} a_j e^{\lambda_j t}\mathbf{v}_j, \tag{1.2}$$

where the scalar coefficients $\{a_j\}_{j=1}^{n}$ are determined by the initial condition $\mathbf{x}(t_0) = \mathbf{x}_0$.

We see from (1.2) that the possible behaviors of $\mathbf{x}(t)$ depend on characteristics of $\sigma(\mathbf{A})$. For example, if all eigenvalues of $\mathbf{A}$ have a negative real part, the state is guaranteed to decay to zero in every component as time increases. We also see that purely imaginary eigenvalues

will generate oscillations that neither decay nor grow. Questions of the system's stability are answered by analyzing the spectrum of the matrix $\mathbf{A}$.

A common generalization of this problem introduces a matrix to the left hand side of the equation (1.1), yielding

$$\mathbf{E}\mathbf{x}'(t) = \mathbf{A}\mathbf{x}(t), \tag{1.3}$$

where $\mathbf{A}, \mathbf{E} \in \mathbb{C}^{n\times n}$. Using the same ansatz as before, we arrive at the *generalized* eigenvalue problem:

$$\lambda\mathbf{E}\mathbf{v} = \mathbf{A}\mathbf{v},$$

or, equivalently

$$(\mathbf{A} - \lambda\mathbf{E})\mathbf{v} = \mathbf{0}.$$

This is referred to as the eigenvalue problem corresponding to the *matrix pencil* $\mathbf{A} - \lambda\mathbf{E}$, also denoted $(\mathbf{A}, \mathbf{E})$. When $\mathbf{E}$ is invertible, the generalized eigenvalue problem is theoretically equivalent to the standard eigenvalue problem for the matrix $\mathbf{E}^{-1}\mathbf{A}$. When $\mathbf{E}$ is singular, we refer to equation (1.3) as a differential-algebraic system of equations, reviewed extensively in [27]. Further examination of the standard eigenvalue problem can be found in [50].

1.2 Higher Degree Linear Dynamical Systems and Polynomial Eigenvalue Problems

Consider the degree d linear differential equation,

$$\sum_{k=0}^{d} \mathbf{A}_k \mathbf{x}^{(k)}(t) = \mathbf{0},$$

where $\mathbf{A}_0, \ldots, \mathbf{A}_d \in \mathbb{C}^{n\times n}$ and $\mathbf{x}^{(k)}(t)$ denotes the k^{th} derivative of $\mathbf{x}(t)$. In addition, second order equations are generated by mass-spring-damper systems, giving them importance when studying vibrations of idealized structures (see [42]). Making the same ansatz as before, $\mathbf{x}(t) = e^{\lambda t}\mathbf{v}$, we come to the polynomial eigenvalue problem:

$$\left(\sum_{k=0}^{d} \lambda^k \mathbf{A}_k\right) \mathbf{v} = \mathbf{0}, .$$

We can analyze this problem by converting to a generalized eigenvalue problem of greater dimension. Consider the quadratic case,

$$(\mathbf{A}_0 + \lambda \mathbf{A}_1 + \lambda^2 \mathbf{A}_2)\mathbf{v} = \mathbf{0}.$$

Defining concatenated matrices, we can pose this as a generalized eigenvalue problem of dimension $2n$,

$$\left(\begin{bmatrix} \mathbf{A}_0 & \mathbf{A}_1 \\ 0 & \mathbf{I} \end{bmatrix} + \lambda \begin{bmatrix} 0 & \mathbf{A}_2 \\ -\mathbf{I} & 0 \end{bmatrix}\right) \begin{bmatrix} \mathbf{v} \\ \lambda \mathbf{v} \end{bmatrix} = \mathbf{0}.$$

A similar procedure for the degree d case yields a dimension dn generalized eigenvalue problem. Here we see a possible numerical complication. The dimension of the problem is compounded by the degree of the differential equation, which is problematic when the order

of the system is very large. This is also significant from the standpoint of approximation capabilities. Later we will discuss replacing general nonlinear eigenvalue problems with polynomial approximations, but this analysis shows that larger degree approximations increase the dimension of the resulting problem considerably.

1.3 Delay Dynamical Systems and a *Truly* Nonlinear Eigenvalue Problem

Although the previous example did yield a nonlinear eigenvalue problem, we showed that it could be recomposed as a generalized linear eigenvalue problem of larger dimension. Now we consider a dynamical system that generates an eigenvalue problem with more exotic features, highlighting the interest (and difficulty) of general nonlinear eigenvalue problems. Consider the delay differential equation

$$\mathbf{x}'(t) = \mathbf{A}\mathbf{x}(t-\tau),$$

where $\mathbf{A} \in \mathbb{C}^{n\times n}$ and $\tau > 0$. Delay differential equations are used for a variety of applications, including describing incubation periods for diseases and transmission delays in networks. The preface of [33] provides a comprehensive overview of the applications of delay systems. Here, the length of the delay enters as a physical parameter denoted by τ. Using the ansatz $\mathbf{x}(t) = e^{\lambda t}\mathbf{v}$ as before, we arrive at the nonlinear eigenvalue problem:

$$\lambda e^{\lambda\tau}\mathbf{v} = \mathbf{A}\mathbf{v}.$$

This last equation implies that the quantity $\lambda e^{\lambda\tau}$ is an eigenvalue of the matrix $\mathbf{A}$, in the linear eigenvalue sense. Taking the scalar case where $\mathbf{A} = 1$ and setting $\tau = 1$, our problem reduces to the scalar nonlinear equation $\lambda e^{\lambda} = 1$. This generates infinitely many possible values for λ given by the multi-valued Lambert-W function, detailed in depth in [16], yielding in this case $\lambda_k = W_k(1)$. Figure 1.1 shows the structure of the set of solutions of $\lambda e^{\lambda} = 1$. The integer k denotes the use of the k^{th} branch of the Lambert-W function, in the same way one may refer to different branches of the complex logarithm. In the case where $\mathbf{A} \in \mathbb{C}^{n\times n}$, with $\sigma(\mathbf{A}) = \{\mu_1, \ldots, \mu_n\}$, then any value taken by $\lambda_{j,k} = W_k(\tau\mu_j)/\tau$ is a solution to this nonlinear eigenvalue problem, and the eigenvectors to the nonlinear problem are simply the corresponding eigenvectors of $\mathbf{A}$.

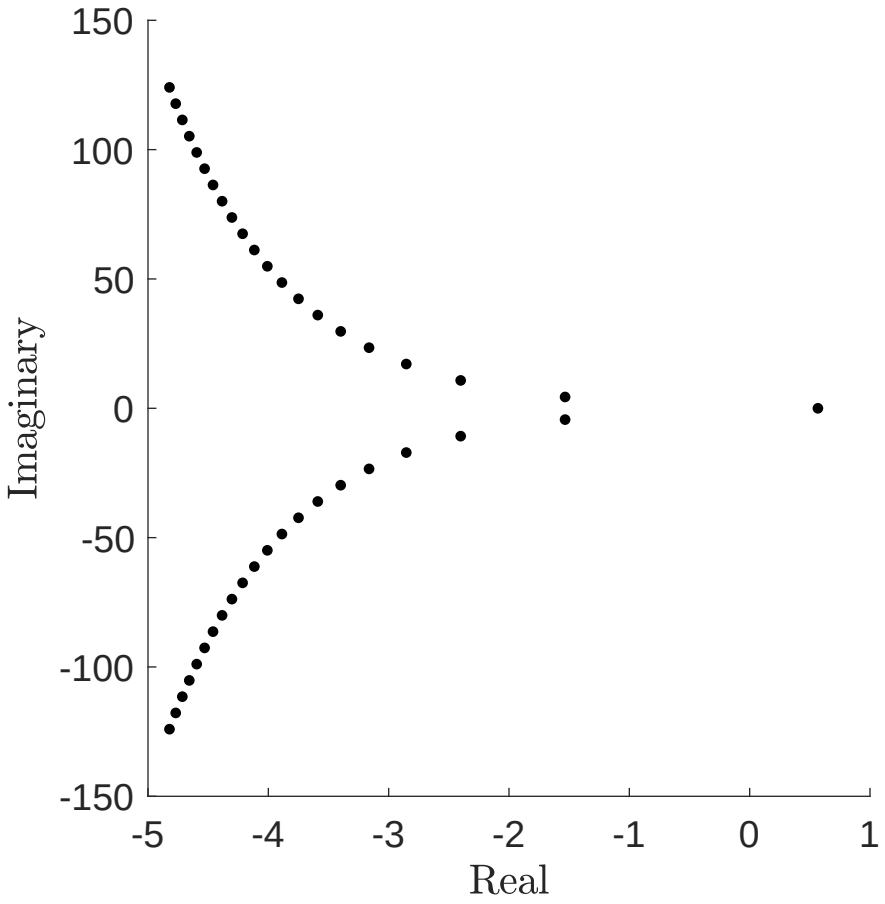

Figure 1.1: $W_k(1)$ for $k = -20, \ldots, 20$.

This example exposes several difficulties with general nonlinear eigenvalue problems. First, their nonlinear structure requires specific treatment. In the case above, the well studied Lambert-W function is available, but this is not the case for all problems. Second, similar to the polynomial case, the dimension of the underlying state compounds an already difficult problem. Here, we had to consider the n eigenvalues of $\mathbf{A}$, but the computation of the spectrum of $\mathbf{A}$ might be burdensome to begin with. Finally, we note that for both

the polynomial and delay eigenvalue problems, we lose linear independence of eigenvectors. Simply by the fact that there are more eigenvalues then n, we see that the set of eigenvectors cannot be linearly independent. In this particular case, all eigenvalues generated by $\lambda_{j,k} = W_k(\tau\mu_j)/\tau$ will have the same eigenvector; the eigenvector of $\mathbf{A}$ corresponding to μ_j.

This simple delay problem exposes several key attributes of nonlinear eigenvalue problems and allows us to specify the locations of eigenvalues and characteristics of eigenvectors with ease. Therefore we will use this problem as the main numerical example for the following numerical methods. Most nonlinear eigenvalue problems do not have such an easy solution. We intend to develop new algorithm to solve such problems. For the interested reader, many more nonlinear eigenvalue problems are described in [13].

1.4 Numerical Experiments

For our main numerical example, we use the simple delay problem described above with $\tau = 1$, defined by

$$\mathbf{T}(\lambda) \coloneqq \lambda\mathbf{I} - e^{-\lambda}\mathbf{A}, \quad \mathbf{T}(\lambda)\mathbf{v} = \mathbf{0},$$

where we set the size of $\mathbf{A}$ to be $n = 100$, with $\sigma(\mathbf{A}) = \{-1, -2, \ldots, -n\}$. We set the left and right eigenvectors of $\mathbf{A}$ using a seeded random number generator. Four branches of exact eigenvalues are plotted in Figure 1.2. In the following chapters, this numerical example will be referred to as Example 1.

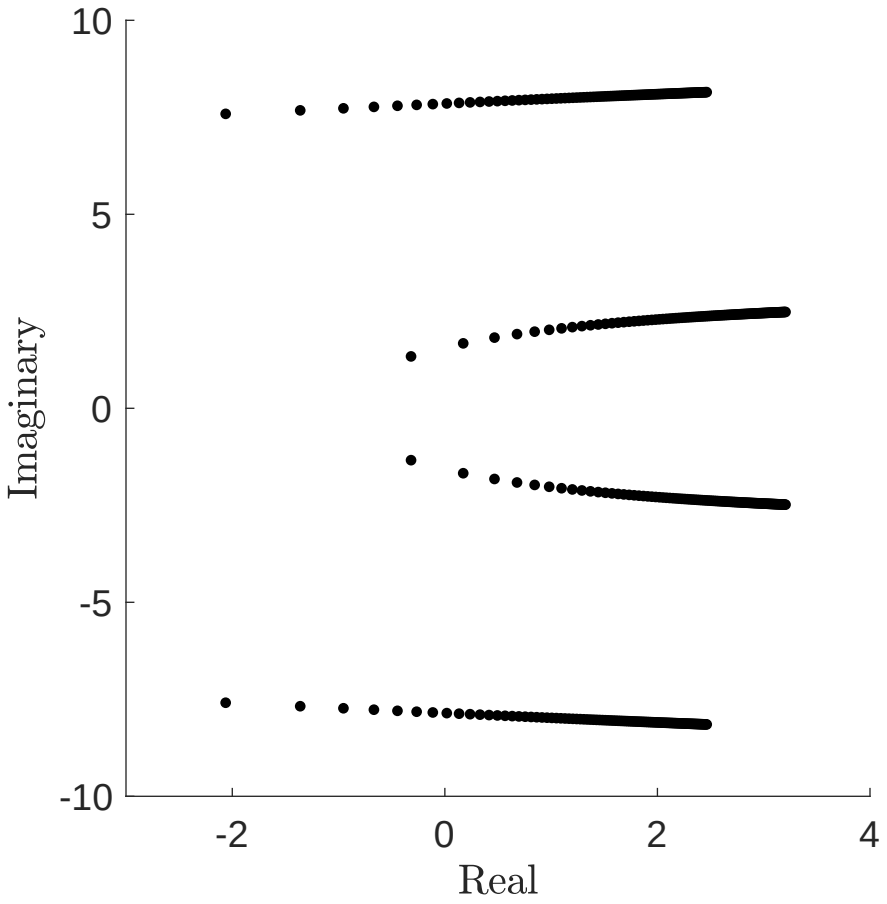

Figure 1.2: Exact Solutions for Example 1: Simple delay problem

1.5 Measuring Error of Approximate Eigenvalue-Eigenvector Pairs

Given that this thesis will introduce several methods for approximating the solutions to nonlinear eigenvalue problems, we need a method of measuring the success of these estimates.

Let $(\lambda, \mathbf{v})$ denote a true eigenpair of the nonlinear eigenvalue problem

$$\mathbf{T}(\lambda)\mathbf{v} = \mathbf{0}$$

that we approximate with the pair $(\theta, \mathbf{w})$, where we assume $\|\mathbf{v}\| = \|\mathbf{w}\| = 1$. There are several ways of measuring the difference between these two pairs. Linear stability analysis for a dynamical system often reduces to finding the locations of the eigenvalues to an underlying nonlinear problem. To this end, it makes sense to measure the distance between the true and approximate eigenvalue, $|\lambda - \theta|$. We will refer to this quantity as the *eigenvalue error*. This measure does not account for accuracy in the eigenvector. To do so, we could measure $\|\mathbf{v} - \mathbf{w}\|$, assuming we have normalized and orientated the two vectors. Alternatively, we can compute the angle between the two vectors to avoid considering normalization. One method that considers the approximate eigenpair together is the residual norm for $(\theta, \mathbf{w})$, meaning $\|\mathbf{T}(\lambda)\mathbf{v} - \mathbf{T}(\theta)\mathbf{w}\| = \|\mathbf{T}(\theta)\mathbf{w}\|$. In the linear case with $\mathbf{T}(\lambda) = \mathbf{A} - \lambda\mathbf{E}$, the residual norm is the familiar quantity $\|(\mathbf{E} - \theta\mathbf{A})\mathbf{w}\|$.

1.6 Contributions of this Thesis

This thesis makes several contributions to current numerical methods for nonlinear eigenvalue problems. The first primary effort goes toward connecting recent contour integration methods to results regarding system identification. In Chapter 4, we show that these methods implicitly apply the Silverman realization algorithm to identify an underlying linear dynamical system related to the set of eigenvalues in a chosen region of the complex plane.

We then expand this methodology by using general rational interpolation for system identification, producing a Loewner matrix contour integration technique. In Chapter 5, we study the application of rational interpolation directly to the function $\mathbf{T}(z)^{-1}$, and we find that the poles of this interpolant approximate the eigenvalues of $\mathbf{T}$. We expand this idea to several iterative methods, where at each step the approximate eigenvalues are taken as new interpolation points. We show that the case where one interpolation point is used is theoretically equivalent to the application of Newton's method for a particular scalar function.

Chapter 2 provides an overview of current methods that relate to the new proposed methods developed in the proceeding chapters. Chapter 3 provides the necessary background for Loewner matrix techniques used in systems theory and model reduction that guide the development of the two new methods proposed in this thesis. Finally Chapter 6 will summarize the conclusions of this work.

Chapter 2

Overview of current methods

In this chapter will overview several methods for approximating solutions to nonlinear eigenvalue problems. We do this to inform ourselves of the strengths and weaknesses of these methods, which will help us evaluate the merits of the new techniques proposed in later chapters. First, we must give our definition of a solved problem. As discussed in Chapter 1, there are several hurdles to solving large scale nonlinear eigenvalue problems, including the large dimension of the problem and its nonlinear structure. For the purpose of this work, we assume that handling either one of these issues results in a simplified problem that can be solved using existing algorithms. For example, we will discuss *linearization* methods where the nonlinear problem is approximated by a generalized linear eigenvalue problem of larger dimension. Because the theory of such generalized eigenvalue problems is well developed, we consider this the end of the treatment of the nonlinear problem and pass the baton accordingly. On the other hand, we will also discuss how one can use *subspace projection* to form an

approximating nonlinear problem of smaller dimension. In this case, our goal is to form an approximate problem small enough to pass to standard techniques for nonlinear equations.

2.1 Linearization Methods

We start with perhaps the most intuitive set of methods, linearization methods. Considering the general nonlinear eigenvalue problem, given as

$$\mathbf{T}(z)\mathbf{v} = 0, \quad \mathbf{T} : \mathbb{C} \to \mathbb{C}^{n\times n},$$

the underlying idea for methods of this type is to approximate the nonlinear terms of $\mathbf{T}$ using a set of approximating functions, typically either polynomials or rational functions. Polynomials and rational functions are commonly used because the underlying theory for solving problems of these types is more developed than for general nonlinear problems (see [29] and [18] for overviews). The following result found in [21] characterizes how an approximate nonlinear eigenvalue problem can be formed by approximating the matrix valued function $\mathbf{T}$ itself.

Theorem 2.1.1

Let $\mathbf{T} : \mathbb{C} \to \mathbb{C}^{n\times n}$ *define the nonlinear eigenvalue problem of interest and consider* $\mathbf{P} : \mathbb{C} \to \mathbb{C}^{n\times n}$*, an approximating function. Assume for the domain* $\Omega \subseteq \mathbb{C}$ *that*

$$\|\mathbf{T} - \mathbf{P}\|_\Omega := \max_{z\in\Omega} \|\mathbf{T}(z) - \mathbf{P}(z)\|_2 < \epsilon$$

for some $\epsilon > 0$*. If* $\mathbf{P}$ *has an eigenvalue* $\lambda \in \Omega$ *with the right eigenvector* $\mathbf{v}$ *normalized so*

that $\|\mathbf{v}\|_2 = 1$*, then*

$$\|\mathbf{T}(\lambda)\mathbf{v}\|_2 < \epsilon.$$

Proof. Given $\lambda \in \sigma(\mathbf{P})$ with unit-length right eigenvector $\mathbf{v}$, we have $\mathbf{P}(\lambda)\mathbf{v} = \mathbf{0}$, and therefore

$$\|\mathbf{T}(\lambda)\mathbf{v}\|_2 = \|(\mathbf{T}(\lambda) - \mathbf{P}(\lambda))\mathbf{v}\|_2 \leq \|\mathbf{T}(\lambda) - \mathbf{P}(\lambda)\|_2\|\mathbf{v}\|_2 = \|\mathbf{T}(\lambda) - \mathbf{P}(\lambda)\|_2 < \epsilon.$$

□

However, it is important to note that enforcing small $\|\mathbf{T}(\lambda)\mathbf{v}\|$ does not necessarily imply that λ is close to an eigenvalue. Take the linear example where

$$\mathbf{T}(z) = \begin{bmatrix} z-\lambda & 1 \\ 0 & z-\lambda \end{bmatrix} \text{ and } \mathbf{P}(z) = \begin{bmatrix} z-\lambda & 1 \\ \epsilon & z-\lambda \end{bmatrix}. \tag{2.1}$$

We have enforced that $\|\mathbf{T}(z) - \mathbf{P}(z)\|_2 = \epsilon$. Note that $\mathbf{T}$ has a repeated eigenvalue at λ, where the eigenvalues of $\mathbf{P}$ are $\lambda \pm \sqrt{\epsilon}$. Scaling this example to the dimension n case, we find that the eigenvalues of $\mathbf{P}$ lie on the circle radius $|\epsilon^{1/n}|$ centered at λ, and therefore the eigenvalues of $\mathbf{T}$ and $\mathbf{P}$ differ by $\epsilon^{1/n}$. Figure 2.1 shows these eigenvalues for the case where $n = 100$, $\epsilon = 10^{-5}$ and $\lambda = 0$, and we see a relatively large error of $|\epsilon^{1/n}| = 0.891$ between the true and approximate eigenvalues.

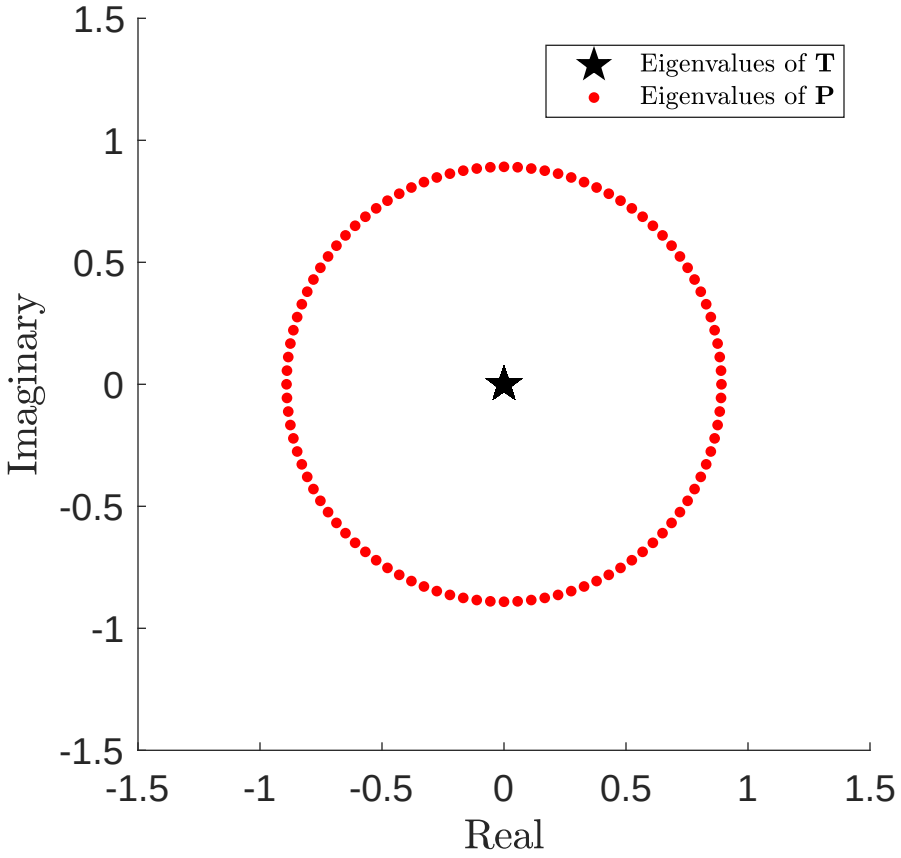

Figure 2.1: Eigenvalues of $\mathbf{T}$ and $\mathbf{P}$ defined in Equation 2.1 with $n = 100$; $\epsilon = 10^{-5}$; $\lambda = 0$

2.1.1 Polynomial Linearization

We now review linearization using polynomials as the approximating functions. Recall from Chapter 1 that a polynomial eigenvalue problem can be converted to a generalized eigenvalue

problem. Given the polynomial eigenvalue problem of degree d and dimension n defined by

$$\left(\sum_{k=0}^{d} \lambda^k \mathbf{A}_k\right) \mathbf{v} = \mathbf{0}, \tag{2.2}$$

where $\mathbf{A}_k \in \mathbb{C}^{n\times n}$ for $k = 0, \ldots, d$, we can form the generalized eigenvalue problem

$$\underbrace{\begin{bmatrix} \mathbf{A}_0 & \mathbf{A}_1 & \cdots & \mathbf{A}_{d-1} \\ \mathbf{0} & \mathbf{I} & \cdots & \mathbf{0} \\ \vdots & & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} \end{bmatrix}}_{\mathbf{B}_0} \underbrace{\begin{bmatrix} \mathbf{v} \\ \lambda\mathbf{v} \\ \vdots \\ \lambda^{d-1}\mathbf{v} \end{bmatrix}}_{\mathbf{w}} = \lambda \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{0} & \cdots & -\mathbf{A}_d \\ \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \ddots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{I} & \mathbf{0} \end{bmatrix}}_{\mathbf{B}_1} \underbrace{\begin{bmatrix} \mathbf{v} \\ \lambda\mathbf{v} \\ \vdots \\ \lambda^{d-1}\mathbf{v} \end{bmatrix}}_{\mathbf{w}},$$

or

$$\mathbf{B}_0\mathbf{w} = \lambda\mathbf{B}_1\mathbf{w},$$

where $\mathbf{B}_0, \mathbf{B}_1 \in \mathbb{C}^{nd\times nd}$ and $\mathbf{w} \in \mathbb{C}^{nd}$. This generalized eigenvalue problem is referred to as a *companion problem* for the polynomial problem defined in (2.2). Other formulations for companion problems exist, and can be attractive when there is structure present in the matrices $\mathbf{A}_j$ that one wishes to exploit. For example, Mehrmann and Watkins [32] detail a *structure-preserving* companion formulation for quadratic eigenvalue problems generated by Hamiltonian dynamical systems. We will focus on the quality of the polynomial approximation to the nonlinear problem, rather than the computation of the eigenvalues of the polynomial problem. In the numerical experiments below, we use Matlab's `polyeig` routine for the computation of eigenvalues and eigenvectors of polynomial eigenvalue problems.

Lagrange Interpolating Polynomials

A natural selection for approximating polynomials is for them to interpolate the original function at a set of interpolation points. That is, provided interpolation points $\{z_j\}_{j=0}^d \subset \mathbb{C}$, we form a degree d matrix valued polynomial $\mathbf{P}$, such that

$$\mathbf{P}(z_j) = \mathbf{T}(z_j)$$

for $j = 0, \ldots, d$. A common framework for achieving this is to introduce Lagrange polynomials for the set of interpolation points.

Definition 2.1.1. Given the set of interpolation points $\{z_j\}_{j=0}^d \subset \mathbb{C}$, we define corresponding Lagrange polynomials ℓ_j for $j = 0, \ldots, d$:

$$\ell_j(z) \coloneqq \prod_{k \neq j} \frac{z - z_k}{z_j - z_k}.$$

Note that each ℓ_j is a polynomial of degree d and that

$$\ell_j(z_m) = \begin{cases} 1, & j = m; \\ 0, & j \neq m. \end{cases}$$

Denoting the interpolation data $\mathbf{T}_j \coloneqq \mathbf{T}(z_j)$, the matrix-valued polynomial defined by

$$\mathbf{P}(z) \coloneqq \sum_{j=0}^{d} \ell_j(z) \mathbf{T}_j,$$

interpolates $\mathbf{T}$ at the set of interpolation points, meaning

$$\mathbf{P}(z_j) = \mathbf{T}(z_j)$$

for $j = 0, \ldots, d$. This process allows us to form an interpolating polynomial for a given nonlinear eigenvalue problem, provided a set of interpolation points. Choices for interpolations points are picked based on a search region. If crude approximations for eigenvalues of interest are already available, these can be considered for interpolation points.

Below we have simulated this situation for the simple delay problem (Example 1) introduced in Chapter 1. We have provided four approximate eigenvalues to be used as interpolation points. After forming the interpolating polynomial, we rewrite the polynomial in the monomial basis, and compute the eigenvalues of the approximate polynomial problem using `polyeig`.

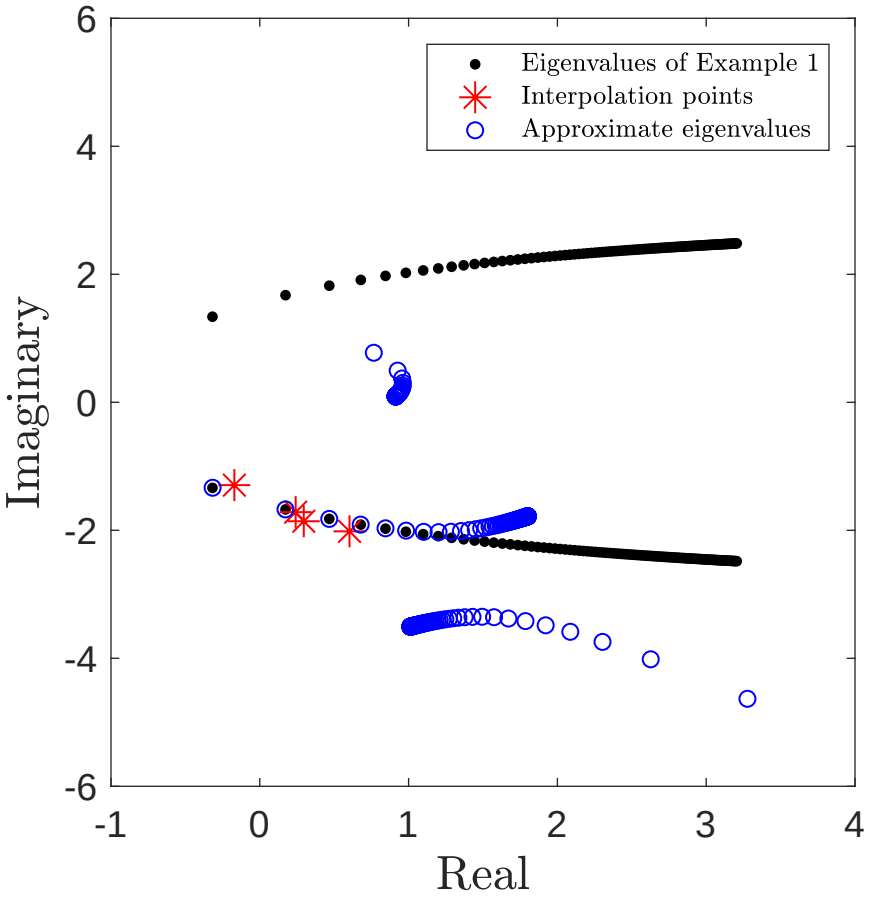

Figure 2.2: Eigenvalues of a Degree 3 Lagrange Polynomial

Given that the dimension of the problem $n = 100$ and we used a degree $d = 3$ approximating polynomial, we obtain 300 approximate eigenvalues. Certainly the four eigenvalues close to the interpolation points are well approximated, with the maximum error of these four being 3.5×10^{-3}. We also see that many of the other eigenvalues in the same branch are well approximated. However, we also see two clusters of eigenvalues of the polynomial problem that do not approximate any true eigenvalue; they are completely spurious. In fact

only seven of the 300 approximate eigenvalue yielded eigenvalue error less than 10^{-1}. This raises the problem of deciding which eigenvalues are successful approximations. For more results using Lagrange interpolation, see [47].

Taylor Polynomials

An another possible choice for an approximating polynomial is to enforce interpolation of the derivatives of $\mathbf{T}$. If we desire all eigenvalues near a particular point $\zeta_0 \in \mathbb{C}$, we can form the Taylor polynomial centered at ζ_0, defined as

$$\mathbf{P}(z) \coloneqq \sum_{j=0}^{d} \frac{\mathbf{T}^{(j)}(\zeta_0)}{j!}(z-\zeta_0)^j,$$

enforcing interpolation of the derivatives of $\mathbf{T}$ at ζ_0 up to order d. For comparison, we have taken the centroid of the interpolation points used in the Lagrange interpolant shown in Figure 2.1.1 to be ζ_0. Similar to the Lagrange interpolation case, we see accurate approximations for eigenvalues near ζ_0. To compare with the Lagrange interpolant, the four eigenvalues closest to the approximation center were still well approximated, with the maximum error 7.7×10^{-3}, but again there were only seven approximate eigenvalues with error less then 10^{-1}.

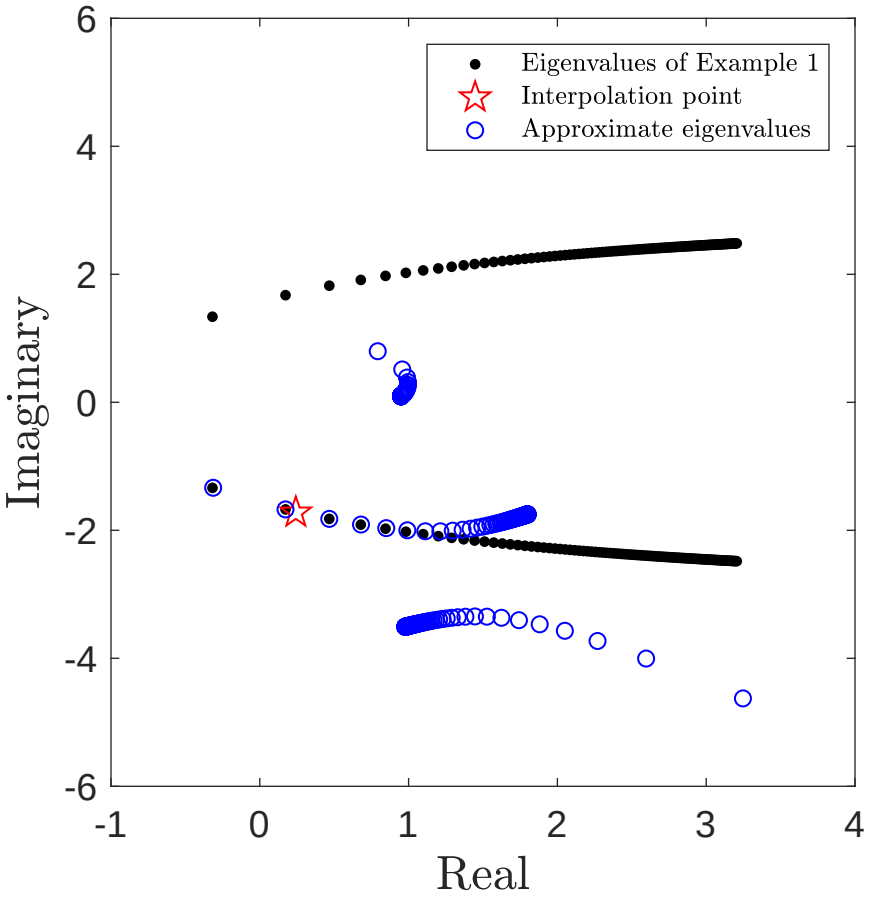

Figure 2.3: Eigenvalues of a Degree 3 Taylor Polynomial

Other types of approximating polynomials have been investigated as well, including Newton and Chebyshev polynomials, which can be found in [1].

2.1.2 Rational Linearization

There has also been work in linearization methods using rational approximation for nonlinear eigenvalue problems. In this case, the polynomial function $\mathbf{P}(z)$ above is replaced by a rational approximant $\mathbf{Q}(z)$, where one enforces $\mathbf{Q} \approx \mathbf{T}$ in a region of interest. Current methodologies include using rational Newton polynomials, where the approximating functions are defined by providing desired roots and poles [22]. The compact rational Krylov (CORK) method developed in [46] provides a framework for using a generalization of the *compact Arnoldi decomposition* to reduce the computation cost of the rational linearization step. There is also a process for using the *adaptive Antoulas-Anderson* (AAA) method for rational approximation found in [30]. The AAA algorithm is developed in [34] with its origins laid down in [6].

2.2 Projection Methods

In contrast to linearization, projection methods look to preserve the nonlinear structure of a problem while reducing the dimension of the problem. One forms a subspace

$$\mathcal{U} = \operatorname{Range} \mathbf{U},$$

where $\mathbf{U} \in \mathbb{C}^{n \times r}$ is sub-unitary, and defines the projected nonlinear problem

$$\mathbf{T}_r(z) \coloneqq \mathbf{U}^* \mathbf{T}(z) \mathbf{U}.$$

We obtain a smaller, but still nonlinear problem to work with, given that $\mathbf{T}(z)$ has dimension $n \times n$, and $\mathbf{T}_r(z)$ has dimension $r \times r$. We assume that the reduction in dimension is sufficient so that one can realistically pass the new problem to standard small-scale nonlinear eigenvalue algorithms. This leaves us with the task of picking the subspace $\mathcal{U}$. Two notable methods for this are the Arnoldi [48] and Jacobi-Davidson methods [14,49], which are inspired by corresponding methods of the same name for the linear eigenvalue problem. Another choice informed by work in structure preserving model reduction (see [11,41]) is to form the subspace $\mathcal{U}$ so that $\mathbf{T}_r(z)^{-1}$ interpolates $\mathbf{T}(z)^{-1}$ at a set of interpolation points $\{\sigma_j\}_{j=1}^r \subset \mathbb{C}$ along directions $\{\mathbf{w}_j\}_{j=1}^r \subset \mathbb{C}^n$, i.e.,

$$\mathbf{T}_r(\sigma_j)^{-1}\mathbf{U}^*\mathbf{w}_j = \mathbf{T}(\sigma_j)^{-1}\mathbf{w}_j,$$

for $j = 1, \ldots, r$. This can be achieved by a specific choice of the projective subspace.

Theorem 2.2.1

Given $\mathbf{T}(z)$, $\{\sigma_j\}_{j=1}^r \subset \mathbb{C}$ *and* $\{\mathbf{w}_j\}_{j=1}^r \subset \mathbb{C}^n$, *let* $\mathbf{U}$ *be constructed as*

$$\mathbf{U} = \operatorname{orth}\left[\mathbf{T}(\sigma_1)^{-1}\mathbf{w}_1 \ \cdots \ \mathbf{T}(\sigma_r)^{-1}\mathbf{w}_r\right].$$

Then the reduced nonlinear problem $\mathbf{T}_r(z)$ *satisfies*

$$\mathbf{T}_r(\sigma_j)^{-1}\mathbf{U}^*\mathbf{w}_j = \mathbf{T}(\sigma_j)^{-1}\mathbf{w}_j, \quad j = 1, \ldots, r$$

An apparent strength of this methodology is that it can be applied iteratively. Starting with an initial selection of interpolation points $\{\sigma_j\}_{j=1}^r$ and directions $\{\mathbf{w}_j\}_{j=1}^r$, we can form the reduced problem $\mathbf{T}_r(z) = \mathbf{U}^*\mathbf{T}(z)\mathbf{U}$ as detailed before, and compute r desired eigenvalues $\{\lambda_j\}_{j=1}^r$ and eigenvectors $\{\mathbf{v}_j\}_{j=1}^r$ of $\mathbf{T}_r$. We then update the interpolation points and

directions using these eigenvalues and eigenvectors. This iteration is summarized in Algorithm 1, taking inspiration from the *dominate pole algorithm* (DPA) [36] and the *iterative rational Krylov algorithm* (IRKA) [20].

Algorithm 1 Iterative Projection

input: initial interpolation points and directions $\{\sigma_j\}_{j=1}^r$ and $\{\mathbf{w}_j\}_{j=1}^r$.

for $k = 0, 1, \ldots$ until converged **do**

1. Form the subspace $\mathbf{U} = \text{orth}\,[\mathbf{T}(\sigma_1)^{-1}\mathbf{w}_1 \ \cdots \ \mathbf{T}(\sigma_r)^{-1}\mathbf{w}_r]$.
2. Form the projected nonlinear eigenvalue problem: $\mathbf{T}_r(z) = \mathbf{U}^*\mathbf{T}(z)\mathbf{U}$.
3. Compute r desired eigenvalues $\{\lambda_j\}_{j=1}^r$ and eigenvectors $\{\mathbf{v}_j\}_{j=1}^r$ of $\mathbf{T}_r$.
4. Update interpolation points and directions: $\sigma_j \leftarrow \lambda_j$, $\mathbf{w}_j \leftarrow \mathbf{U}\mathbf{v}_j$.
5. **end for**

Figure 2.4 shows the results of this method applied to the simple delay equation of Example 1, where we have taken $r = 2$. The two initial points were placed just right of the two principle branches in hopes of converging to the right-most eigenvalues. We see from the zoomed-in plot on the right that the approximations converged to eigenvalues slightly in the interior of the spectrum. The average error measured by the Euclidean distance between the approximate and true eigenvalues shown here was 4.46×10^{-4}. In this case, both interpolation points converged to the same point, and thus only one approximation appears per branch of solutions.

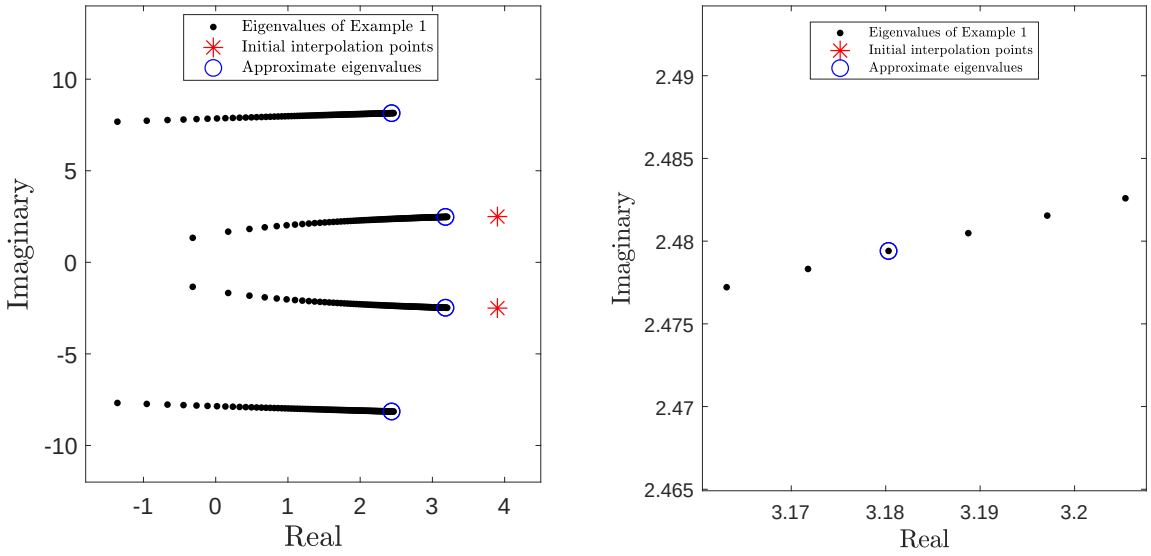

Figure 2.4: Results of Iterative Projection

2.3 Contour Integration Methods

The following methods rely on contour integration in the complex plane to extract a linear problem that shares a set of eigenvalues in a search region. This is made possible using a theorem from Mstislav Keldysh detailed in [23,24], where we can guarantee a particular form for the inverse of a matrix valued function; the theorem explicitly tells us how the eigenvalues and eigenvectors of $\mathbf{T}$ enter into a Laurent decomposition of $\mathbf{T}(z)^{-1}$. We will refer to this decomposition as the Keldysh decomposition for $\mathbf{T}(z)^{-1}$. We provide the theorem for the case where the eigenvalues of interest are semi-simple, meaning the algebraic multiplicity of the eigenvalues matches the number of corresponding linearly independent eigenvectors.

Theorem 2.3.1 (Keldysh [23, 24])

Let $\mathbf{T} : \mathbb{C} \to \mathbb{C}^{n\times n}$ *be analytic in the domain* $\Omega \subseteq \mathbb{C}$*. Assume that* $\mathbf{T}$ *has* m *eigenvalues denoted* $\lambda_1, \ldots, \lambda_m$ *in* Ω*, counting multiplicities. If each eigenvalue is semi-simple, then*

$$\mathbf{T}(z)^{-1} = \mathbf{V}(z\mathbf{I} - \mathbf{\Lambda})^{-1}\mathbf{W}^* + \mathbf{R}(z) = \sum_{j=1}^{m} \frac{\mathbf{v}_j \mathbf{w}_j^*}{z - \lambda_j} + \mathbf{R}(z),$$

where $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_m]$, $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_m] \in \mathbb{C}^{n\times m}$ *contain the right and left eigenvectors satisfying the normalization condition that* $\mathbf{w}_j^* \mathbf{T}'(\lambda_j)\mathbf{v}_j = 1$, $\mathbf{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_m)$*, and* $\mathbf{R} : \mathbb{C} \to \mathbb{C}^{n\times n}$ *is analytic in* Ω*.*

The key fact here is that information regarding the eigenvalues in Ω is captured in the summation term of the right hand side. Given that $\mathbf{R}$ is analytic, this term integrates to zero over any closed path contained in Ω by Cauchy's Theorem. The methods detailed by Asakura, Sakurai, Tadano, Ikegami and Kimura in [10] and Beyn in [15] use this fact to extract the nonlinear eigenvalues from a linear eigenvalue problem. To present this clearly, we first introduce the Cauchy integral theorem, which is used later.

Theorem 2.3.2 (Cauchy's Integral Theorem and Formula)

Let $f : \Omega \to \mathbb{C}$ *be analytic on the simply connected set* Ω*. If* Γ *is a rectifiable closed path in* Ω*, then*

$$\int_\Gamma f(z)dz = 0,$$

and

$$f^{(n)}(w) = \frac{n!}{2\pi i} \int_\Gamma \frac{f(z)}{(z-w)^{n+1}} dz.$$

These two facts lead us to the following result, which is stated in a more general case in [15] where the eigenvalues are permitted to be derogatory.

Theorem 2.3.3

Let $f : \Omega \to \mathbb{C}$ be analytic, and consider the nonlinear eigenvalue problem for $\mathbf{T}$ with the Keldysh decomposition given in Theorem 2.3.1. Then

$$\frac{1}{2\pi i}\int_\Gamma f(z)\mathbf{T}(z)^{-1}dz = \mathbf{V}f(\mathbf{\Lambda})\mathbf{W}^*.$$

Proof. Given that f and $\mathbf{R}$ are analytic in Ω, we have

$$\frac{1}{2\pi i}\int_\Gamma f(z)\mathbf{R}(z)dz = 0$$

by the Cauchy integral theorem. Therefore we have that

$$\begin{aligned}\frac{1}{2\pi i}\int_\Gamma f(z)\mathbf{T}(z)^{-1}\,dz &= \frac{1}{2\pi i}\int_\Gamma f(z)\mathbf{V}(z\mathbf{I}-\mathbf{\Lambda})^{-1}\mathbf{W}^*\,dz + \frac{1}{2\pi i}\int_\Gamma f(z)\mathbf{R}(z)\,dz \\ &= \frac{1}{2\pi i}\int_\Gamma f(z)\mathbf{V}(z\mathbf{I}-\mathbf{\Lambda})^{-1}\mathbf{W}^*\,dz.\end{aligned}$$

Computing this integral with

$$\mathbf{V}(z\mathbf{I}-\mathbf{\Lambda})^{-1}\mathbf{W}^* = \sum_{j=1}^{m}\frac{\mathbf{v}_j\mathbf{w}_j^*}{z-\lambda_j},$$

we obtain

$$\begin{aligned}\frac{1}{2\pi i}\int_\Gamma f(z)\mathbf{T}(z)^{-1}\,dz &= \frac{1}{2\pi i}\int_\Gamma f(z)\sum_{j=1}^{m}\frac{\mathbf{v}_j\mathbf{w}_j^*}{z-\lambda_j}\,dz \\ &= \sum_{j=1}^{m}\mathbf{v}_j\left(\frac{1}{2\pi i}\int_\Gamma\frac{f(z)}{z-\lambda_j}\,dz\right)\mathbf{w}_j^* \\ &= \sum_{j=1}^{m}\mathbf{v}_j f(\lambda_j)\mathbf{w}_j^* \\ &= \mathbf{V}f(\mathbf{\Lambda})\mathbf{W}^*.\end{aligned}$$

□

We now describe two methods that are detailed in [15], and are closely related to methods developed in [10]. First, consider the case where the number of eigenvalues in the search region Ω is less than the dimension of the problem, i.e., $m \leq n$, and when their corresponding eigenvectors are linearly independent. In the linear case, eigenvectors corresponding to distinct eigenvalues are always linear independent, but this condition places a restriction on the search region for general nonlinear eigenvalue problems. Note that then the matrices $\mathbf{V}$ and $\mathbf{W}$ are rank m, as their columns are m linearly independent eigenvectors of $\mathbf{T}$. We first introduce a probing matrix $\mathbf{\Theta} \in \mathbb{C}^{n \times r}$, where $r < n$, and consider the quantity

$$\mathbf{A}_f = \frac{1}{2\pi i} \int_\Gamma f(z)\mathbf{T}(z)^{-1}\mathbf{\Theta}\, dz \in \mathbb{C}^{n \times r}.$$

The purpose of this probing matrix is to reduce the computational burden of evaluating $\mathbf{T}(z)^{-1}$ during the numerical evaluation of the contour integral. Now consider two choices for the function f as $f(z) = 1$ and $f(z) = z$, defining

$$\mathbf{A}_0 \coloneqq \frac{1}{2\pi i} \int_\Gamma \mathbf{T}(z)^{-1}\mathbf{\Theta}\, dz, \quad \mathbf{A}_1 \coloneqq \frac{1}{2\pi i} \int_\Gamma z\, \mathbf{T}(z)^{-1}\mathbf{\Theta}\, dz. \tag{2.3}$$

Then by Theorem 2.3.3, we have

$$\mathbf{A}_0 = \mathbf{V}\mathbf{W}^*\mathbf{\Theta}, \quad \mathbf{A}_1 = \mathbf{V}\mathbf{\Lambda}\mathbf{W}^*\mathbf{\Theta}. \tag{2.4}$$

We now state results from [15], which produces a method for extracting the eigenvalues $\lambda_1, \ldots, \lambda_m$.

Theorem 2.3.4

Suppose that $\mathbf{\Theta}$ *is chosen such that* $\mathbf{W}^*\mathbf{\Theta} \in \mathbb{C}^{m\times r}$ *has rank* m*. Defining* $\mathbf{A}_0$ *and* $\mathbf{A}_1$ *as in (2.3), compute the short SVD of* $\mathbf{A}_0$*:*

$$\mathbf{A}_0 = \mathbf{V}_0\mathbf{\Sigma}_0\mathbf{W}_0^*, \tag{2.5}$$

where $\mathbf{V}_0 \in \mathbb{C}^{n\times m}$*,* $\mathbf{\Sigma} = \operatorname{diag}(\sigma_1, \ldots, \sigma_m)$*, and* $\mathbf{W}_0 \in \mathbb{C}^{r\times m}$*. Then the eigenvalues of*

$$\mathbf{M} \coloneqq \mathbf{V}_0^*\mathbf{A}_1\mathbf{W}_0\mathbf{\Sigma}_0^{-1}$$

are the eigenvalues of $\mathbf{T}$ *contained in* Ω*, i.e.,* $\lambda_1, \ldots, \lambda_m$*. Also, the corresponding eigenvectors of* $\mathbf{T}$*, denoted* $\mathbf{v}_k$ *for* $k = 1, \ldots, m$ *can be computed as*

$$\mathbf{v}_k = \mathbf{V}_0\mathbf{q}_k,$$

where $\mathbf{q}_k$ *are the eigenvectors of* $\mathbf{M}$ *corresponding to* λ_k*.*

This theorem relies on the rank condition that $\mathbf{\Theta}$ is chosen so that $\mathbf{W}^*\mathbf{\Theta}$ is rank m, which implies that $\operatorname{Range}\mathbf{A}_0 = \operatorname{Range}\mathbf{V} = \operatorname{Range}\mathbf{V}_0$. This theorem also sheds light on where this procedure fails in the case where the number of enclosed eigenvalues is greater then the dimension of the problem, i.e., $m > n$. In this case, the rank of $\mathbf{W}^*\mathbf{\Theta}$ must be less than m. We now summarize section 5 of [15], which details how the case where $m > n$ (when the eigenvectors of the eigenvalues contained in Ω are linear dependent) can be treated. Similar to the definitions of $\mathbf{A}_0$ and $\mathbf{A}_1$, we define

$$\mathbf{A}_p = \frac{1}{2\pi i}\int_\Gamma z^p\ \mathbf{T}(z)^{-1}\ \mathbf{\Theta}\ dz,$$

noting that by Theorem 2.3.3, we have

$$\mathbf{A}_p = \mathbf{V}\mathbf{\Lambda}^p\mathbf{W}^*\mathbf{\Theta}. \tag{2.6}$$

We now choose an integer $K \geq 1$, and build the *Hankel* and *shifted Hankel* matrices, denoted $\mathbf{H}_0$ and $\mathbf{H}_1$ respectively:

$$\mathbf{H}_0 = \begin{bmatrix} \mathbf{A}_0 & \mathbf{A}_1 & \cdots & \mathbf{A}_{K-1} \\ \mathbf{A}_1 & \mathbf{A}_2 & \cdots & \mathbf{A}_K \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{K-1} & \mathbf{A}_K & \cdots & \mathbf{A}_{2K-2} \end{bmatrix}, \quad \mathbf{H}_1 = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 & \cdots & \mathbf{A}_K \\ \mathbf{A}_2 & \mathbf{A}_3 & \cdots & \mathbf{A}_{K+1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_K & \mathbf{A}_{K+1} & \cdots & \mathbf{A}_{2K-1} \end{bmatrix}.$$

Note that $K = 1$ yields $\mathbf{H}_0 = \mathbf{A}_0$ and $\mathbf{H}_1 = \mathbf{A}_1$. Given the decomposition for $\mathbf{A}_p$ given in (2.6), we can decompose the Hankel matrices as

$$\mathbf{H}_0 = \widetilde{\mathbf{V}}\widetilde{\mathbf{W}}^*, \quad \mathbf{H}_1 = \widetilde{\mathbf{V}}\mathbf{\Lambda}\widetilde{\mathbf{W}}^*, \tag{2.7}$$

where the matrices $\widetilde{\mathbf{V}}$ and $\widetilde{\mathbf{W}}$ are defined as

$$\widetilde{\mathbf{V}} = \begin{bmatrix} \mathbf{V} \\ \hline \mathbf{V}\mathbf{\Lambda} \\ \hline \vdots \\ \hline \mathbf{V}\mathbf{\Lambda}^{K-1} \end{bmatrix}, \quad \widetilde{\mathbf{W}}^* = \left[\mathbf{W}^*\mathbf{\Theta} \,\middle|\, \mathbf{\Lambda}\mathbf{W}^*\mathbf{\Theta} \,\middle|\, \cdots \,\middle|\, \mathbf{\Lambda}^{K-1}\mathbf{W}^*\mathbf{\Theta}\right]. \tag{2.8}$$

We assume that we have chosen K such that the rank of $\widetilde{\mathbf{W}}$ is m. We can then compute the eigenvalues of interest in a similar fashion to Theorem 2.3.4.

Theorem 2.3.5

Suppose that $\mathbf{\Theta}$ *is chosen such that* $\widetilde{\mathbf{W}}$ *has rank* m*. Then we compute the short SVD of* $\mathbf{H}_0$

to be

$$\mathbf{H}_0 = \mathbf{V}_0 \mathbf{\Sigma}_0 \mathbf{W}_0^*, \tag{2.9}$$

where $\mathbf{V}_0 \in \mathbb{C}^{Kn \times m}$, $\mathbf{\Sigma} = \text{diag}(\sigma_1, \ldots, \sigma_m)$, *and* $\mathbf{W}_0 \in \mathbb{C}^{Kr \times m}$. *Then the eigenvalues of*

$$\mathbf{M} = \mathbf{V}_0^* \mathbf{H}_1 \mathbf{W}_0 \mathbf{\Sigma}_0^{-1}$$

are the eigenvalues of $\mathbf{T}$ *contained in* Ω, *i.e.,* $\lambda_1, \ldots, \lambda_m$. *Also, the corresponding eigenvectors of* $\mathbf{T}$, *denoted* $\mathbf{v}_k$ *for* $k = 1, \ldots, m$, *can be computed as*

$$\mathbf{v}_k = \mathbf{V}_0^{(1)} \mathbf{q}_k,$$

where $\mathbf{q}_k$ *are the eigenvectors of* $\mathbf{M}$ *corresponding to* λ_k, *and* $\mathbf{V}_0^{(1)}$ *is the upper* $n \times m$ *block of* $\mathbf{V}_0$,

$$\mathbf{V}_0 = \left[\begin{array}{c} \mathbf{V}_0^{(1)} \\ \hline \mathbf{V}_0^{(2:K)} \end{array} \right].$$

2.3.1 Integral Approximation Using the Trapezoid Rule

To perform these contour integration methods, we must approximate contour integrals of the form

$$\mathbf{A}_f = \frac{1}{2\pi i} \int_{\partial\Omega} f(z) \mathbf{T}(z)^{-1} \, dz$$

with a numerical quadrature rule. As in [15], we assume that the boundary of Ω, denoted $\partial\Omega = \Gamma$, has a smooth and 2π-periodic parameterization, meaning that $\phi \in C^1$ and $\phi(\theta +$

$2\pi) = \phi(\theta)$ for $\theta \in \mathbb{R}$. This leads to

$$\mathbf{A}_f = \frac{1}{2\pi i} \int_0^{2\pi} f(\phi(\theta))\mathbf{T}(\phi(\theta))^{-1}\phi'(\theta)\, d\theta.$$

Using a numerical quadrature rule leads to the approximation

$$\widehat{\mathbf{A}}_f = \sum_{j=0}^{N-1} w_j f(\phi(z_j))\mathbf{T}(\phi(z_j))^{-1}\phi'(z_j),$$

where $\{z_j\}_{j=0}^{N-1}$ denotes the set of quadrature points and $\{w_j\}_{j=0}^{N-1}$ denotes the set of quadrature weights. Given the success of the trapezoid rule for periodic functions discussed in [43], the trapezoid rule is a good candidate for performing the numerical integration. Taking N equidistant points along the unit circle $z_j = 2\pi j/N$ for $j = 0, \ldots, N-1$, we obtain weights $w_j = x_j/N$ and the integral approximation

$$\widehat{\mathbf{A}}_f = \frac{1}{iN} \sum_{j=0}^{N-1} f(\phi(z_j))\mathbf{T}(\phi(z_j))^{-1}\phi'(z_j).$$

2.3.2 Numerical Results

Figure 2.5 provides results for the simple delay problem of Example 1 when two full branches of solutions are contained in the region Ω. In this came, $m = 2n$, and thus we must use Theorem 2.3.5. Given that the dimension of the problem, $n = 100$, is relatively small, we do not use a probing matrix. We take $K = 3$, yielding Hankel matrices of size 300. Note that as N increases, the singular value decay of $\mathbf{H}_0$ drops more sharply after 200 singular values. On the right, we plot the geometric mean, maximum and minimum of the residual errors the approximate eigenvalues as N increases. The eigenvectors were normalized so that $\|\mathbf{v}_j\|_2 = 1$.

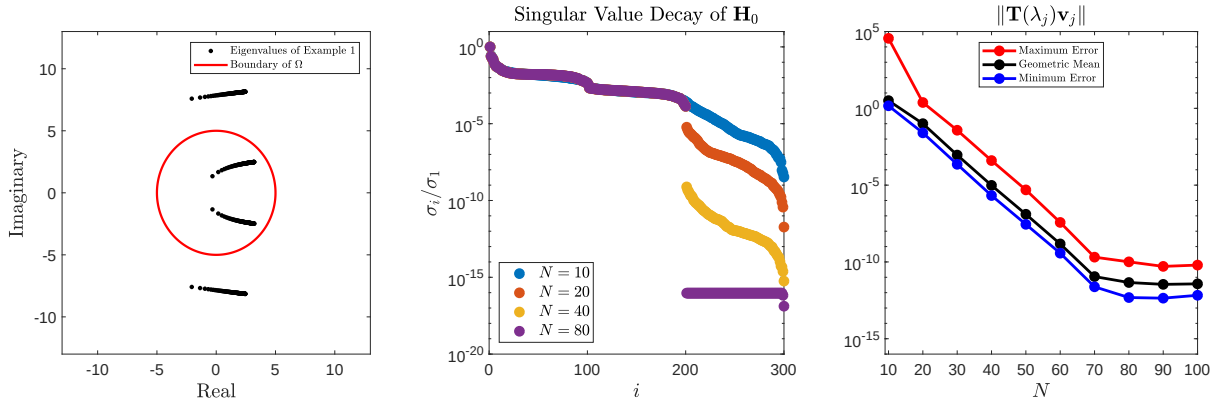

Figure 2.5: (left) Placement of the contour $\partial\Omega$ and the eigenvalues of $\mathbf{T}$; (center) Singular value decays of the Hankel matrix $\mathbf{H}_0$ for varying number of contour points (N); (right) Summary of the residual errors for the approximate eigenvalues.

2.4 Newton's Method

We now detail several ways one can utilize Newton's method for nonlinear eigenvalue problems. These methods are most suitable for computing a single eigenvalue at a time. These methods break into two general categories:

1. One can find a scalar function $f : \mathbb{C} \to \mathbb{C}$ that has roots at some or all of the eigenvalues of $\mathbf{T}$, and then apply Newton's method for scalar functions.

2. One can apply Newton's method for vector valued functions to $\mathbf{T}(\lambda)\mathbf{v}$, enforcing normalization conditions on the eigenvector $\mathbf{v}$.

2.4.1 Scalar Methods

Newton-Trace

A natural choice of a scalar function that has roots at the eigenvalues of $\mathbf{T}$ is the characteristic function, $f(z) = \det(\mathbf{T}(z))$, detailed in Section 5 of [29]. To form the Newton's method iteration, we obtain the derivative of f using Jacobi's formula, yielding

$$f'(z) = \det(\mathbf{T}(z)) \operatorname{Trace}(\mathbf{T}(z)^{-1}\mathbf{T}'(z)).$$

Thus in this case the Newton iteration is defined by

$$z_{j+1} = z_j - \frac{1}{\operatorname{Trace}(\mathbf{T}(z_j)^{-1}\mathbf{T}'(z_j))},$$

where the initial guess z_0 must be provided.

We note that the evaluation of $\mathbf{T}(z_j)^{-1}\mathbf{T}'(z_j)$ requires $O(n^3)$ operations per iteration, restricting the use of this method to modestly sized problems. Figure 2.6 shows the convergence trajectory for the simple delay problem, where the initial guess differs from the closest eigenvalue by 0.1.

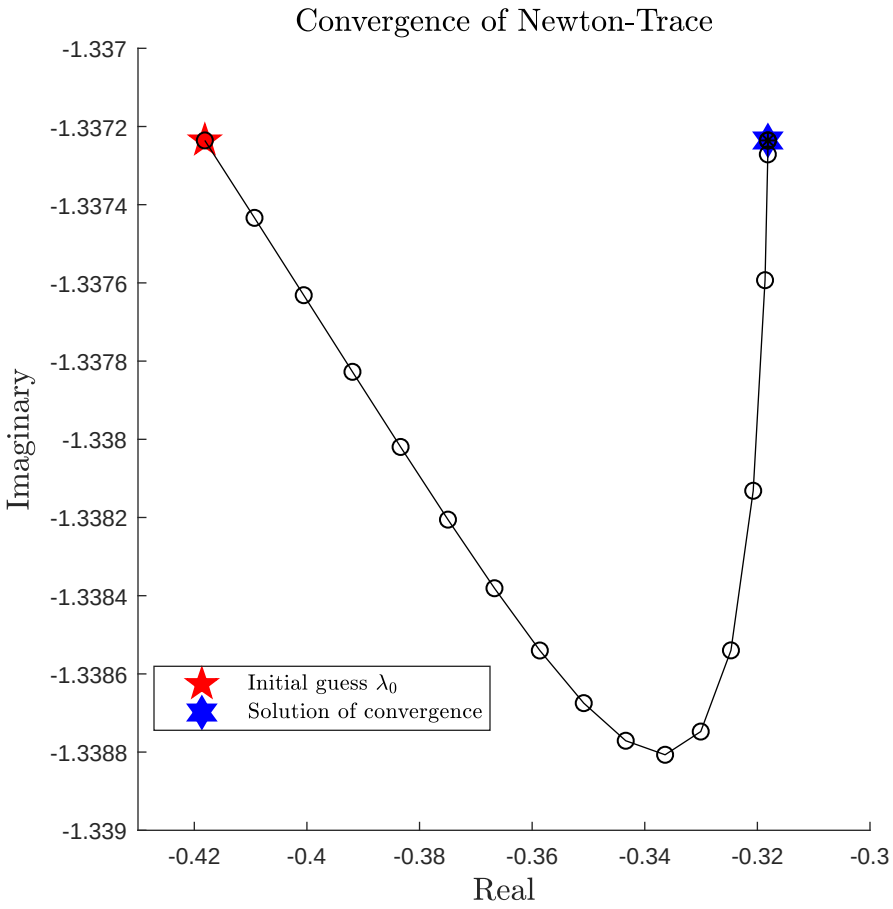

Figure 2.6: Convergence path for the Newton-Trace method.

Newton-QR

Another choice of scalar function comes from exploiting particular matrix factorizations that reveal the rank of the matrix. For example, if one forms the QR decomposition of $\mathbf{T}(z)$ at a particular z, i.e.

$$\mathbf{T}(z) = \mathbf{Q}(z)\mathbf{R}(z),$$

then as z approaches an eigenvalue, one or more of the diagonal entries of $\mathbf{R}(z)$ will approach zero. The Newton-**QR** method was first investigated in [25, 26] and then expanded in [17]. Assume we have

$$\mathbf{T}(z)\mathbf{\Pi}(z) = \mathbf{Q}(z)\mathbf{R}(z),$$

where $\mathbf{\Pi}(z)$ is a permutation matrix enforcing decreasing magnitude diagonal entries in the upper triangular matrix $\mathbf{R}(z)$. In a neighborhood of a simple eigenvalue the permutation matrix will be constant, $\mathbf{\Pi}(z) = \mathbf{\Pi}$. Partitioning the matrix $\mathbf{R}(z)$ as

$$\mathbf{R}(z) = \begin{bmatrix} \mathbf{R}_{1,1}(z) & \mathbf{r}_{1,n}(z) \\ 0 & r_{n,n}(z) \end{bmatrix},$$

it is shown in [17] that

$$r_{n,n}(z) = r_{n,n}(z_0) + \mathbf{e}_n^*\mathbf{Q}(z_0)^*\mathbf{T}'(z_0)\mathbf{\Pi}\mathbf{y}(z - z_0) + O(|z - z_0|^2),$$

where $\mathbf{e}_j$ is the unit vector with 1 in the j^{th} entry. Therefore we can express the derivative term as

$$r'_{n,n}(z) = \mathbf{e}_n^*\mathbf{Q}(z_0)^*\mathbf{T}'(z_0)\mathbf{\Pi}\mathbf{y},$$

thus forming the iteration

$$z_{j+1} = z_j - \frac{r_{n,n}(z_j)}{\mathbf{e}_n^*\mathbf{Q}(z_j)^*\mathbf{T}'(z_j)\mathbf{\Pi}\mathbf{y}}.$$

Similar to the Newton-Trace iteration, this method has a computational cost $O(n^3)$ operations per iteration given the required QR factorization at each step. This cost is greatly reduced for particular structured problems, as shown in [17] for problems with banded structure.

Implicit Determinant

Note that as the Newton-Trace and Newton-QR methods converge to an eigenvalue, we are faced with computations of a near singular matrix $\mathbf{T}(z_j)$. *Bordering, Deletion and Substitution* (BDS) methods look to reduce possible numerical errors emerging from this. We now describe the *Implicit Determinant* method given in [3]. First define a bordered matrix-valued function $\mathbf{G} : \mathbb{C} \to \mathbb{C}^{(n+1)\times(n+1)}$:

$$\mathbf{G}(z) := \begin{bmatrix} \mathbf{T}(z) & \mathbf{b} \\ \mathbf{c}^* & 0 \end{bmatrix},$$

where $\mathbf{b}, \mathbf{c} \in \mathbb{C}^n$ are picked so that $\mathbf{G}(z)$ is nonsingular and ideally well conditioned at a simple eigenvalue λ. As shown in [2], taking $\mathbf{c}^*\mathbf{v} \neq 0$ and $\mathbf{w}^*\mathbf{b} \neq 0$, where $\mathbf{v}$ and $\mathbf{w}$ are the right and left eigenvectors of $\mathbf{T}$ for the eigenvalue λ ensures that $\mathbf{G}(\lambda)$ is nonsingular. For a vector $\mathbf{x} \in \mathbb{C}^n$ such that $\mathbf{c}^*\mathbf{x} = 1$, define the linear system

$$\begin{bmatrix} \mathbf{T}(z) & \mathbf{b} \\ \mathbf{c}^* & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ f \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

By Cramer's rule, we have that

$$f(z) = \frac{\det(\mathbf{T}(z))}{\det(\mathbf{G}(z))},$$

and $f(\lambda) = 0$ if and only if $\lambda \in \sigma(\mathbf{T})$. Differentiating, we obtain

$$\begin{bmatrix} \mathbf{T}(z) & \mathbf{b} \\ \mathbf{c}^* & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}' \\ f' \end{bmatrix} = \begin{bmatrix} -\mathbf{T}'(z)\mathbf{x}'(z) \\ 0 \end{bmatrix}.$$

One then solves both linear systems using a factorization of $\mathbf{G}(z_j)$ to obtain $f(z_j)$ and $f'(z_j)$, and applies Newton's method accordingly. Table 2.1 summarizes convergence results for the

Table 2.1: Convergence summary for scalar newton methods applied to Example 1

Method	Trace	QR	Implicit Determinant
Iterations	17	3	5

three scalar Newton's method routines. Each method was performed with the same initial guess z_0, shown in Figure 2.6.

2.4.2 Vector Methods

Newton's method can also be applied to nonlinear eigenvalue problems to obtain both an eigenvalue and a corresponding eigenvector. Noting that $\mathbf{T}(\lambda)\mathbf{v} = \mathbf{0}$ enforces n equations for $n+1$ unknowns, we add a normalization condition on the eigenvector $\mathbf{v}$, namely that $\mathbf{u}_k^*\mathbf{v} = 1$ for a chosen vector $\mathbf{u}_k \in \mathbb{C}^n$. Note that we have allowed for the normalization condition to be updated after each iteration, yielding the dependence on the step number k. This leads to the root finding problem

$$\mathcal{N}(\mathbf{v}, \lambda) = \begin{bmatrix} \mathbf{T}(\lambda)\mathbf{v} \\ \\ \mathbf{u}_k^*\mathbf{v} - 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \\ 0 \end{bmatrix}. \tag{2.10}$$

Applying Newton's method to (2.10) yields the iteration

$$\begin{bmatrix} \mathbf{v}_{k+1} \\ \\ \lambda_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_k \\ \\ \lambda_k \end{bmatrix} - \mathbf{J}_{\mathcal{N}}\left(\mathbf{v}_k, \lambda_k\right)^{-1} \mathcal{N}(\mathbf{v}_k, \lambda_k), \tag{2.11}$$

where $\mathbf{J}_{\mathcal{N}}$ denotes the Jacobian matrix for $\mathcal{N}$. In this case we have

$$\mathbf{J}_{\mathcal{N}}\left(\begin{bmatrix}\mathbf{v}_k\\ \lambda_k\end{bmatrix}\right) = \begin{bmatrix}\mathbf{T}(\lambda_k) & \mathbf{T}'(\lambda_k)\mathbf{v}_k\\ \mathbf{u}_k^* & 0\end{bmatrix}.$$

We can then rewrite this as

$$\mathbf{T}(\lambda_k)\mathbf{v}_{k+1} = (\lambda_k - \lambda_{k+1})\mathbf{T}'(\lambda_k)\mathbf{v}_k \tag{2.12}$$

$$\mathbf{u}_k^*\mathbf{v}_{k+1} = 1. \tag{2.13}$$

We now detail two possible choices for $\mathbf{u}_k$, which lead to the *nonlinear inverse iteration* and the *two sided Rayleigh iteration*.

Inverse Iteration

First, we can decide to fix $\mathbf{u}_k = \mathbf{u}$, a constant vector. In this case, equation (2.11) gives us the nonlinear inverse iteration detailed in [4] and [37], given in Algorithm 2.

Algorithm 2 Nonlinear Inverse Iteration

input: initial pair $(\lambda_0, \mathbf{v}_0)$ with $\|\mathbf{v}_0\| = 1$ and $\mathbf{u} \neq \mathbf{0}$.

for $k = 0, 1, \ldots$ until converged **do**

1. Solve $\mathbf{T}(\lambda_k)\widetilde{\mathbf{v}} = \mathbf{T}'(\lambda_k)\mathbf{v}_k$ for $\widetilde{\mathbf{v}}$.
2. Set $\lambda_{k+1} = \lambda_k - \dfrac{\mathbf{u}^*\mathbf{v}_k}{\mathbf{u}^*\widetilde{\mathbf{v}}}$
3. Set $\mathbf{v}_{k+1} = \dfrac{\widetilde{\mathbf{v}}}{\|\widetilde{\mathbf{v}}\|}$.

end for

Note that by enforcing $\mathbf{u}^*\mathbf{v} = 1$, one avoids convergence to any eigenvector orthogonal to $\mathbf{u}$. Therefore a common method is to pick $\mathbf{u}$ orthogonal to the span of previously computed eigenvectors to avoid convergence to already known solutions [4]. However, recall that for the simple delay problem given in Chapter 1, distinct eigenvalues can share eigenvectors exactly. In this case, choosing $\mathbf{u}$ in this manner will avoid convergence to all eigenvalues that share the previously computed eigenvector.

Two Sided Rayleigh Iteration

Now we consider the case where we update the vector $\mathbf{u}_k$. Taking $\mathbf{u}_k = \mathbf{T}(\lambda_k)\mathbf{w}_k$, where $\mathbf{w}_k$ is an approximate left eigenvector leads us to the *generalized Rayleigh quotient iteration* developed for polynomial eigenvalue problems in [28] and [29]. In this case one can express the eigenvalue update step in Algorithm 2 as

$$\lambda_{k+1} = \lambda_k - \frac{\mathbf{w}_k^*\mathbf{T}(\lambda_k)\mathbf{v}_k}{\mathbf{w}_k^*\mathbf{T}'(\lambda_k)\mathbf{v}_k}.$$

Later this was extended to general nonlinear problems in [38] and [39] as the *Rayleigh functional iteration*, where the update to the approximate eigenvalue is performed by solving

$$\mathbf{w}_{k+1}^*\mathbf{T}(\lambda_{k+1})\mathbf{v}_{k+1} = 0.$$

Algorithm 3 Rayleigh Functional Iteration

input: initial triple $(\lambda_0, \mathbf{v}_0, \mathbf{w}_0)$, with $\|\mathbf{v}_0\| = \|\mathbf{w}_0\| = 1$.

for $k = 0, 1, \ldots$ until converged **do**

1. Solve $\mathbf{T}(\lambda_k)\widetilde{\mathbf{v}} = \mathbf{T}'(\lambda_k)\mathbf{v}_k$ for $\widetilde{\mathbf{v}}$.
2. Set $\mathbf{v}_{k+1} = \dfrac{\widetilde{\mathbf{v}}}{\|\widetilde{\mathbf{v}}\|}$.
3. Solve $\mathbf{T}(\lambda_k)^*\widetilde{\mathbf{w}} = \mathbf{T}'(\lambda_k)^*\mathbf{w}_k$ for $\widetilde{\mathbf{w}}$.
4. Set $\mathbf{w}_{k+1} = \dfrac{\widetilde{\mathbf{w}}}{\|\widetilde{\mathbf{w}}\|}$.
5. Solve $\mathbf{w}_{k+1}^*\mathbf{T}(\lambda_{k+1})\mathbf{v}_{k+1} = 0$ for λ_{k+1}, taken as the closest solution to λ_k if multiple solutions exist.

end for

It should be noted that a factorization of $\mathbf{T}(\lambda_k)$ can be used to solve the linear system involving $\mathbf{T}(\lambda_k)^*$.

Residual Inverse Iteration

The inverse iteration requires a factorization of $\mathbf{T}(\lambda_k)$ at each iteration step, which unfortunately cannot be re-used as λ_k varies. In fact, replacing λ_k in this computation with some constant $\sigma \in \mathbb{C}$ results in an iteration that will not converge to an eigenvalue of $\mathbf{T}$ unless $\mathbf{T}$ is linear. In [35] this problem is avoided by considering the residual between consecutive

approximate eigenvectors. If $\mathbf{T}$ is twice differentiable, then Algorithm 2 can be expressed as

$$\mathbf{v}_k - \mathbf{v}_{k+1} = \mathbf{v}_k - (\lambda_k - \lambda_{k+1})\mathbf{T}(\lambda_k)^{-1}\mathbf{T}'(\lambda_k)\mathbf{v}_k \tag{2.14}$$

$$= \mathbf{T}(\lambda_k)^{-1}\left[\mathbf{T}(\lambda_k) - (\lambda_k - \lambda_{k+1})\mathbf{T}'(\lambda_k)\right]\mathbf{v}_k \tag{2.15}$$

$$= \mathbf{T}(\lambda_k)^{-1}\mathbf{T}(\lambda_k)\mathbf{v}_k + O(|\lambda_k - \lambda_{k+1}|^2). \tag{2.16}$$

Considering only the linear terms leads to the iteration

$$\mathbf{v}_{k+1} = \mathbf{v}_k - \mathbf{T}(\lambda_k)^{-1}\mathbf{T}(\lambda_{k+1})\mathbf{v}_k.$$

Note that for this iteration, an update for the approximate eigenvalue λ_{k+1} is required. In [35] it is show that replacing λ_k with a fixed σ does not destroy convergence, which leads to the *Residual inverse iteration*.

Algorithm 4 Residual Inverse Iteration

input: initial pair $(\lambda_0, \mathbf{v}_0)$ with $\|\mathbf{v}_0\| = 1$, $\mathbf{u} \neq \mathbf{0}$ and $\sigma \in \mathbb{C}$:

for $k = 0, 1, \ldots$ until converged **do**

1. Solve $\mathbf{u}^*\mathbf{T}(\sigma)^{-1}\mathbf{T}'(\lambda_{k+1})\mathbf{v}_k = 0$ for λ_{k+1}, where λ_{k+1} is taken as the closest solution to λ_k if multiple solutions exist.
2. Solve $\mathbf{T}(\sigma)\mathbf{x} = \mathbf{T}(\lambda_{k+1})\mathbf{v}_k$ for $\mathbf{x}$.
3. Set $\mathbf{v}_{k+1} = \dfrac{\widetilde{\mathbf{v}}}{\mathbf{u}^*\widetilde{\mathbf{v}}}$, where $\widetilde{\mathbf{v}} = \mathbf{v}_k - \mathbf{x}$.

end for

The convergence behavior for these methods depends on the initial eigenvector $\mathbf{v}_0$. For example, taking a random initial guess for inverse iteration can lead to convergence to an

eigenvalue relatively far away from the initial guess λ_0, as shown in Figure 2.7

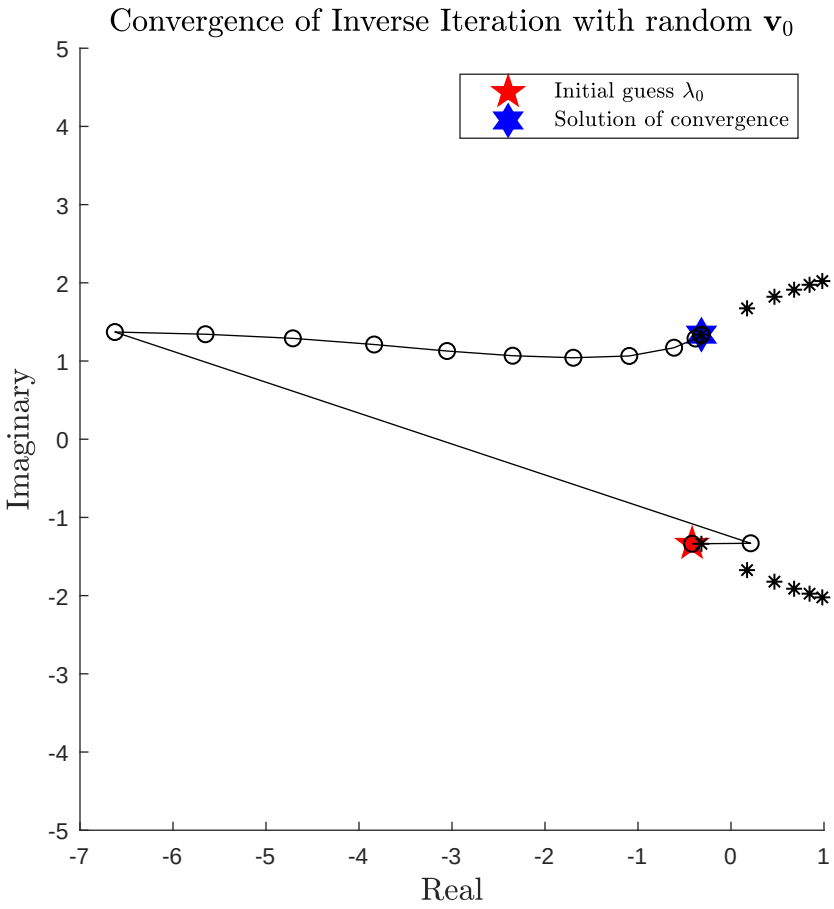

Figure 2.7: Inverse iteration with random $\mathbf{v}_0$ for 15 iterations; λ_0 was a distance 0.1 from the eigenvalue directly right of it.

When $\mathbf{v}_0$ is taken as an approximate eigenvector for λ_0, i.e. , setting $\mathbf{v}_0$ to the last right singular vector of $\mathbf{T}(\lambda_0)$, we obtain convergence to the close eigenvalue in 3 iterations, shown in Figure 2.8.

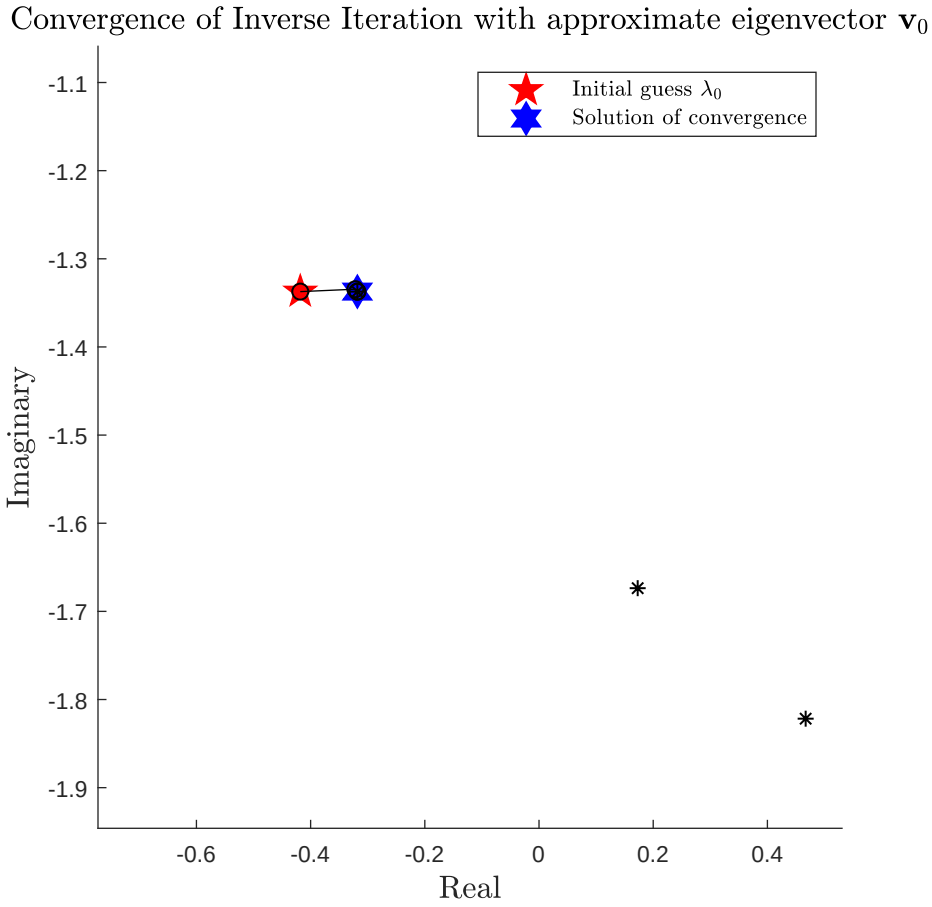

Figure 2.8: Inverse Iteration with approximate eigenvector $\mathbf{v}_0$ for 3 iterations; λ_0 was a distance 0.1 from the eigenvalue directly right of it.

Chapter 3

The Loewner Framework

In the context of model reduction, Loewner matrix techniques are used for data driven approximation and for realizing the transfer function of linear [31], bilinear and quadratic dynamical systems [8, 19]. In this chapter, we state the definitions of the Loewner ($\mathbb{L}$) and shifted Loewner ($\mathbb{L}_s$) matrices, summarize the Loewner framework for rational interpolation and system realization, and provide useful factorizations for these matrices. Then we briefly describe two ways that Loewner matrix techniques can be used for approximating the solutions to nonlinear eigenvalues problems, which are developed fully in the following chapters.

3.1 Interpolation Theorems

Let

$$\mathbf{H} : \mathbb{C} \to \mathbb{C}^{n\times n}$$

be a matrix valued function with possibly countably infinitely many poles in $\mathbb{C}$. The following definitions and theorems summarize rational interpolation of $\mathbf{H}$ using the Loewner framework and can be found in [9], along with more information on Loewner matrix methods in model reduction.

Theorem 3.1.1

Given distinct interpolation points partitioned into two sets:

$$\{z_j\}_{j=1}^{2r} = \{\sigma_j\}_{j=1}^{r} \cup \{\mu_j\}_{j=1}^{r} \subset \mathbb{C},$$

and interpolation directions partitioned into two sets:

$$\{\mathbf{d}_j\}_{j=1}^{2r} = \{\boldsymbol{\ell}_j\}_{j=1}^{r} \cup \{\mathbf{r}_j\}_{j=1}^{r} \subset \mathbb{C}^n,$$

assume that $\mathbf{H}(z)$ is defined for all $z \in \{\sigma_j\}_{j=1}^{r} \cup \{\mu_j\}_{j=1}^{r}$. Define vectors of left and right data:

$$\mathbf{b}_j = \mathbf{H}(\sigma_j)^*\boldsymbol{\ell}_j, \quad \mathbf{c}_j = \mathbf{H}(\mu_j)\mathbf{r}_j,$$

for $j = 1, \ldots r$ and matrices of left and right data:

$$\mathbf{B}_r = [\,\mathbf{b}_1, \ldots, \mathbf{b}_r\,]^* \in \mathbb{C}^{r\times n}, \quad \mathbf{C}_r = [\,\mathbf{c}_1, \ldots, \mathbf{c}_r\,] \in \mathbb{C}^{n\times r}.$$

Also define the Loewner ($\mathbb{L} \in \mathbb{C}^{r\times r}$) and shifted Loewner ($\mathbb{L}_s \in \mathbb{C}^{r\times r}$) matrices:

$$(\mathbb{L})_{j,k} = \frac{\boldsymbol{\ell}_j^* \left[\mathbf{H}(\sigma_j) - \mathbf{H}(\mu_k)\right] \mathbf{r}_k}{\sigma_j - \mu_k} = \frac{\mathbf{b}_j^*\mathbf{r}_k - \boldsymbol{\ell}_j^*\mathbf{c}_k}{\sigma_j - \mu_k},$$

$$(\mathbb{L}_s)_{j,k} = \frac{\boldsymbol{\ell}_j^* \left[\sigma_j \mathbf{H}(\sigma_j) - \mu_k \mathbf{H}(\mu_k)\right] \mathbf{r}_k}{\sigma_j - \mu_k} = \frac{\sigma_j \mathbf{b}_j^* \mathbf{r}_k - \mu_k \boldsymbol{\ell}_j^* \mathbf{c}_k}{\sigma_j - \mu_k},$$

for $j, k = 1, \ldots, r$. *If* $\det(\mathbb{L}_s - s\mathbb{L}) \neq 0$ *for* $s \in \{\sigma_j\}_{j=1}^r \cup \{\mu_j\}_{j=1}^r$, *then the rational matrix valued function*

$$\mathbf{G}(z) \coloneqq \mathbf{C}_r \left(\mathbb{L}_s - z\mathbb{L}\right)^{-1} \mathbf{B}_r$$

satisfies the tangential interpolation conditions:

$$\boldsymbol{\ell}_j^* \mathbf{G}(\sigma_j) = \boldsymbol{\ell}_j^* \mathbf{H}(\sigma_j), \quad \mathbf{G}(\mu_j)\mathbf{r}_j = \mathbf{H}(\mu_j)\mathbf{r}_j,$$

for $j = 1, \ldots, r$. *If* $\sigma_j = \mu_j$, *for* $j = 1, \ldots, r$, *then* $\mathbf{G}(z)$ *also satisfies the Hermite interpolation condition*

$$\boldsymbol{\ell}_j \mathbf{G}'(\sigma_j)\mathbf{r}_j = \boldsymbol{\ell}_j \mathbf{H}'(\sigma_j)\mathbf{r}_j,$$

where the the diagonal entries of the Loewner matrices are defined

$$(\mathbb{L})_{jj} = \boldsymbol{\ell}_j^* \mathbf{H}'(\sigma_j)\mathbf{r}_j, \quad (\mathbb{L}_s)_{jj} = \boldsymbol{\ell}_j^* \left(\sigma_j \mathbf{H}'(\sigma_j) + \mathbf{H}(\sigma_j)\right) \mathbf{r}_j$$

for $j = 1, \ldots, r$.

This interpolation result allows for *data driven* approximation, as even when the specific structure of $\mathbf{H}(z)$ is not known, we can form this rational interpolant using measurements of $\mathbf{H}(z)$ at the interpolation points. In the context of linear dynamical systems however, the Loewner framework provides a methodology for identifying a dynamical system. Consider the linear dynamical system

$$\mathbf{E}\mathbf{x}'(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \quad \mathbf{A}, \mathbf{E} \in \mathbb{C}^{n\times n}, \mathbf{B} \in \mathbb{C}^{n\times m}$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t), \quad \mathbf{C} \in \mathbb{C}^{p\times n}, \mathbf{D} \in \mathbb{C}^{p\times m},$$

which yields the *transfer function*

$$\mathbf{H}(z) = \mathbf{C}(z\mathbf{E} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}. \tag{3.1}$$

The transfer function is ubiquitous in the field of dynamical systems, as it maps the input $\mathbf{u}$ to the output $\mathbf{y}$ in the frequency domain, i.e.,

$$\widehat{\mathbf{y}}(z) = \mathbf{H}(z)\widehat{\mathbf{u}}(z),$$

where $\widehat{\mathbf{y}}$ and $\widehat{\mathbf{u}}$ are the Laplace transforms of $\mathbf{y}$ and $\mathbf{u}$. We see that the poles of $\mathbf{H}(z)$ are in fact the eigenvalues of the matrix pencil $(\mathbf{A}, \mathbf{E})$. The Loewner matrices for this transfer function $\mathbf{H}(z)$ have useful factorizations into the *generalized observability* matrix $\mathcal{O}$, and the *generalized reachability* matrix $\mathcal{R}$.

Definition 3.1.1. Consider the transfer function $\mathbf{H}(z)$ defined in (3.1), and left/right interpolation points $\{\sigma_j\}_{j=1}^r, \{\mu_j\}_{j=1}^r \subset \mathbb{C}$, and left/right directions $\{\boldsymbol{\ell}_j\}_{j=1}^r, \{\mathbf{r}_j\}_{j=1}^r \subset \mathbb{C}^n$. Assume that $\mathbf{H}(z)$ is defined at each interpolation point. Then we define

$$\mathcal{O} = \begin{bmatrix} \boldsymbol{\ell}_1^*\mathbf{C}(\sigma_1\mathbf{E} - \mathbf{A})^{-1} \\ \vdots \\ \boldsymbol{\ell}_r^*\mathbf{C}(\sigma_r\mathbf{E} - \mathbf{A})^{-1} \end{bmatrix}, \text{ and } \mathcal{R} = \begin{bmatrix} (\mu_1\mathbf{E} - \mathbf{A})^{-1}\mathbf{B}\mathbf{r}_1, & \cdots & , (\mu_r\mathbf{E} - \mathbf{A})^{-1}\mathbf{B}\mathbf{r}_r \end{bmatrix},$$

where $\mathcal{O}$ is referred to as the *generalized observability matrix* and $\mathcal{R}$ is referred to as the *generalized reachability matrix* for the transfer function $\mathbf{H}(z)$.

Theorem 3.1.2

The Loewner matrices for the transfer function $\mathbf{H}(z)$ *defined in (3.1) have the factorizations*

$$\mathbb{L} = -\mathcal{O}\mathbf{E}\mathcal{R}$$

$$\mathbb{L}_s = -\mathcal{O}\mathbf{A}\mathcal{R}.$$

We include a proof for the Hermite interpolation case of this factorization for completeness, and because several algebraic steps provide insight into computations performed in the following chapters. The Lagrange case can be recovered using the first step of the proof. These factorizations are also given in [7] and [31], where they are discussed in further detail.

Proof. First we show that the off-diagonal entries of $\mathbb{L}$ indeed match those of $-\mathcal{O}\mathbf{E}\mathcal{R}$. We have

$$\begin{aligned}(\mathbb{L})_{jk} &= \frac{\boldsymbol{\ell}_j^* \left[\mathbf{H}(\sigma_j) - \mathbf{H}(\sigma_k)\right] \mathbf{r}_k}{\sigma_j - \sigma_k} \\ &= \frac{\boldsymbol{\ell}_j^* \mathbf{C} \left[(\sigma_j \mathbf{E} - \mathbf{A})^{-1} - (\sigma_k \mathbf{E} - \mathbf{A})^{-1}\right] \mathbf{B}\mathbf{r}_k}{\sigma_j - \sigma_k},\end{aligned}$$

using the definition of $\mathbf{H}$ and canceling out the $\mathbf{D}$ terms. Factoring the resolvent terms on either side, we obtain for $j \neq k$

$$\begin{aligned}(\mathbb{L})_{jk} &= \frac{\boldsymbol{\ell}_j^* \mathbf{C} (\sigma_j \mathbf{E} - \mathbf{A})^{-1} \left[(\sigma_k \mathbf{E} - \mathbf{A}) - (\sigma_j \mathbf{E} - \mathbf{A})\right] (\sigma_k \mathbf{E} - \mathbf{A})^{-1} \mathbf{B}\mathbf{r}_k}{\sigma_j - \sigma_k} \\ &= \frac{\boldsymbol{\ell}_j^* \mathbf{C} (\sigma_j \mathbf{E} - \mathbf{A})^{-1} \left[\sigma_k - \sigma_j\right] \mathbf{E} (\sigma_k \mathbf{E} - \mathbf{A})^{-1} \mathbf{B}\mathbf{r}_k}{\sigma_j - \sigma_k} \\ &= -\boldsymbol{\ell}_j^* \mathbf{C} (\sigma_j \mathbf{E} - \mathbf{A})^{-1} \mathbf{E} (\sigma_k \mathbf{E} - \mathbf{A})^{-1} \mathbf{B}\mathbf{r}_k \\ &= -\operatorname{row}_j(\mathcal{O}) \mathbf{E} \operatorname{col}_k(\mathcal{R}) = -(\mathcal{O}\mathbf{E}\mathcal{R})_{jk}.\end{aligned}$$

We also have

$$\begin{aligned}(\mathbb{L})_{jj} &= \boldsymbol{\ell}_j^* \mathbf{H}'(\sigma_j)\mathbf{r}_j \\ &= -\boldsymbol{\ell}_j^* \mathbf{C}(\sigma_j \mathbf{E} - \mathbf{A})^{-1}\mathbf{E}(\sigma_j \mathbf{E} - \mathbf{A})^{-1}\mathbf{B}\mathbf{r}_j \\ &= -\operatorname{row}_j(\mathcal{O})\mathbf{E}\operatorname{col}_j(\mathcal{R}) = -(\mathcal{O}\mathbf{E}\mathcal{R})_{jj}.\end{aligned}$$

Therefore, $\mathbb{L} = -\mathcal{O}\mathbf{E}\mathcal{R}$. Similar steps show that the shifted Loewner matrix can be expressed as $\mathbb{L}_s = -\mathcal{O}\mathbf{A}\mathcal{R}$. □

3.2 The Loewner Matrix Pencil

These factorizations allow us to express the matrix pencil $(\mathbb{L}_s, \mathbb{L})$, which we will call the *Loewner pencil*, as

$$z\mathbb{L} - \mathbb{L}_s = -\mathcal{O}(z\mathbf{E} - \mathbf{A})\mathcal{R}.$$

We can investigate three cases relating the eigenvalues of the Loewner pencil $(\mathbb{L}_s, \mathbb{L})$ and the poles of the transfer function $\mathbf{H}$, which are given by the eigenvalues of the matrix pencil $(\mathbf{A}, \mathbf{E})$.

1. If the rank of either $\mathcal{O}$ or $\mathcal{R}$ are less then n, one can view $z\mathbb{L} - \mathbb{L}_s$ as a compression of $z\mathbf{E} - \mathbf{A}$. In this case, $\mathbf{G}(z)$ is a lower order interpolant of $\mathbf{H}(z)$. We will not recover the poles of $\mathbf{H}(z)$ from the Loewner pencil.

2. If $\operatorname{rank}\mathcal{O} = \operatorname{rank}\mathcal{R} = r = n$, then $\mathcal{O}$ and $\mathcal{R}$ are square and invertible, and the eigenvalues of the Loewner pencil $(\mathbb{L}_s, \mathbb{L})$ are the poles of the transfer function $\mathbf{H}(z)$;

we recover the poles exactly. In fact, we recover the transfer function, obtaining $\mathbf{G} = \mathbf{H}$.

3. If $\operatorname{rank}\mathcal{O} = \operatorname{rank}\mathcal{R} = n$ and $r > n$, then the Loewner pencil contains repetitive information. Shown in [31] and [9], we can perform an SVD truncation of the Loewner pencil to obtain a regular realization of $\mathbf{H}$.

Theorem 3.2.1 (SVD truncation)

Assume that

$$\operatorname{rank}\left[\mathbb{L}_s - s\mathbb{L}\right] = \operatorname{rank}[\mathbb{L} \quad \mathbb{L}_s] = \operatorname{rank}\begin{bmatrix} \mathbb{L} \\ \\ \mathbb{L}_s \end{bmatrix} = n$$

for all $s \in \{\sigma_j\}_{j=1}^{r} \cup \{\mu_j\}_{j=1}^{r}$. *Then consider the reduced SVDs of concatenations of* $\mathbb{L}$ *and* $\mathbb{L}_s$, *obtaining*

$$[\mathbb{L} \quad \mathbb{L}_s] = \mathbf{Y}\mathbf{\Sigma}_1\widetilde{\mathbf{X}}^*, \quad \begin{bmatrix} \mathbb{L} \\ \\ \mathbb{L}_s \end{bmatrix} = \widetilde{\mathbf{Y}}\mathbf{\Sigma}_2\mathbf{X}^*$$

where $\widetilde{\mathbf{Y}}, \widetilde{\mathbf{X}}, \in \mathbb{C}^{2r\times n}$, $\mathbf{Y}, \mathbf{X}, \in \mathbb{C}^{r\times n}$, *and* $\mathbf{\Sigma}_1, \mathbf{\Sigma}_2 \in \mathbb{C}^{n\times n}$. *Defining matrices,*

$$\widetilde{\mathbb{L}} = \mathbf{Y}^*\mathbb{L}\mathbf{X}, \quad \widetilde{\mathbb{L}}_s = \mathbf{Y}^*\mathbb{L}_s\mathbf{X}, \quad \widetilde{\mathbf{B}}_r = \mathbf{Y}^*\mathbf{B}, \quad \widetilde{\mathbf{C}}_r = \mathbf{C}_r\mathbf{X},$$

then the transfer function

$$\widetilde{\mathbf{G}}(z) = \widetilde{\mathbf{C}}_r\left(\widetilde{\mathbb{L}}_s - z\widetilde{\mathbb{L}}\right)^{-1}\widetilde{\mathbf{B}}_r$$

achieves the same Lagrange or Hermite interpolation conditions as $\mathbf{G}$, *and is minimal, meaning the matrix-pencil* $(\widetilde{\mathbb{L}}_s, \widetilde{\mathbb{L}})$ *is regular. In this case,* $\widetilde{\mathbf{G}} = \mathbf{H}$.

In theory the concatenation of the Loewner matrices will have n nonzero singular values. In practice one must define the reduced SVDs based on numerical rank.

3.3 Applications to Nonlinear Eigenvalue Problems

Now we look to apply the Loewner framework to the nonlinear eigenvalue problem

$$\mathbf{T}(\lambda)\mathbf{v} = \mathbf{0}.$$

The following chapters propose two general ways of doing this.

1. We can form a similar contour integration method discussed in Chapter 2 and in [10] and [15], where instead of forming the Hankel and shifted Hankel matrices for a particular dynamical system, we form the Loewner pencil. This tool is derived from the Loewner framework's use as a realization tool.

2. We can use the poles of a rational interpolant of $\mathbf{T}(z)^{-1}$ to approximate eigenvalues of the nonlinear problem. We see that in particular cases, this method is equivalent to the application of Newton's method to a particular scalar function.

Chapter 4

Contour Integration in the Loewner Framework

4.1 Connecting Contour Integration with System Identification

This chapter proposes a new contour integration approach for nonlinear eigenvalue problems. Rather than forming the Hankel and shifted Hankel matrices as in [10, 15], our new method is based in system identification via rational interpolation and the Loewner framework.

Assume that $\mathbf{T}$ has m eigenvalues denoted $\lambda_1, \ldots, \lambda_m$ in the domain $\Omega \subseteq \mathbb{C}$, counting multiplicities, and that each eigenvalue is semi-simple. Using the Keldysh decomposition

stated in Theorem 2.3.1, we can express

$$\mathbf{T}(z)^{-1} = \mathbf{V}(z\mathbf{I} - \mathbf{\Lambda})^{-1}\mathbf{W}^* + \mathbf{R}(z) = \sum_{k=1}^{m} \frac{\mathbf{v}_k \mathbf{w}_k^*}{z - \lambda_k} + \mathbf{R}(z), \tag{4.1}$$

where $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_m]$, $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_m]$ are right and left eigenvectors of $\mathbf{T}$ satisfying the normalization condition that $\mathbf{w}_k^* \mathbf{T}'(\lambda_k)\mathbf{v}_k = 1$, $\mathbf{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_m)$, and $\mathbf{R} : \mathbb{C} \to \mathbb{C}^{n \times n}$ is analytic in Ω.

Note that the sum on the right hand side of (4.1) is in fact the transfer function defined in (3.1), with $\mathbf{E} = \mathbf{I}$, $\mathbf{A} = \mathbf{\Lambda}$, $\mathbf{C} = \mathbf{V}$, $\mathbf{B} = \mathbf{W}^*$ and $\mathbf{D} = \mathbf{0}$. We will denote

$$\mathbf{H}(z) := \mathbf{V}(z\mathbf{I} - \mathbf{\Lambda})^{-1}\mathbf{W}^* = \sum_{k=1}^{m} \frac{\mathbf{v}_k \mathbf{w}_k^*}{z - \lambda_k} \tag{4.2}$$

for this reason. We now see that the contour integration methods discussed in Chapter 2 can be thought of as applications of the *Silverman realization algorithm* developed by Silverman in [40] and discussed in [5].

Definition 4.1.1. The *infinite Hankel matrix* $\mathcal{H}$ for the transfer function $\mathbf{H}(z)$ is the block Hankel matrix of *Markov parameters* or moments of a dynamical system. For the system defining the transfer function (4.2), $\mathcal{H}$ is defined

$$\mathcal{H} := \begin{bmatrix} \mathbf{V}\mathbf{W}^* & \mathbf{V}\mathbf{\Lambda}\mathbf{W}^* & \cdots \\ \mathbf{V}\mathbf{\Lambda}\mathbf{W}^* & \mathbf{V}\mathbf{\Lambda}^2\mathbf{W}^* & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}.$$

The Silverman realization algorithm (Lemma 4.41 of [5]) provides a realization for these

Markov parameters. By this we mean a methodology for forming state space matrices that produce the same Markov parameters.

Theorem 4.1.1

Let m be the rank of $\mathcal{H}$. First, find an $m \times m$ sub-matrix $\mathbb{H}$ of $\mathcal{H}$ that has full rank. Then construct the following matrices:

- $\mathbb{H}_s \in \mathbb{C}^{m\times m}$*: a sub-matrix of* $\mathcal{H}$ *that has rows with the same index as* $\mathbb{H}$ *and columns obtained by shifting each column of* $\mathbb{H}$ *by one block.*
- $\Psi \in \mathbb{C}^{m\times n}$*: composed of the same rows as* $\mathbb{H}$*; its columns are the first* m *columns of* $\mathcal{H}$*.*
- $\Phi \in \mathbb{C}^{n\times m}$*: composed of the same columns as* $\mathbb{H}$*; its rows are the first* n *rows of* $\mathcal{H}$*.*

Then

$$\mathbf{G}(z) \coloneqq \Phi(z\mathbf{I} - \mathbb{H}^{-1}\mathbb{H}_s)^{-1}\mathbb{H}^{-1}\Psi = \Phi(z\mathbb{H} - \mathbb{H}_s)^{-1}\Psi = \mathbf{H}(z),$$

meaning we can identify the underlying transfer function exactly.

In the case of the Hankel contour integration methods of [10, 15], rather than finding a $m \times m$ sub-matrix of $\mathcal{H}$, one truncates after a certain number of block columns and rows,

forming

$$\mathbf{H}_0 = \begin{bmatrix} \mathbf{A}_0 & \mathbf{A}_1 & \cdots & \mathbf{A}_{K-1} \\ \mathbf{A}_1 & \mathbf{A}_2 & \cdots & \mathbf{A}_K \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{K-1} & \mathbf{A}_K & \cdots & \mathbf{A}_{2K-2} \end{bmatrix}.$$

The resulting matrices are not necessarily full rank, which is why a SVD truncation based on numerical rank is required. Once the transfer function $\mathbf{H}(z)$ is identified, finding the resulting eigenvalues and eigenvectors can be achieved by diagonalizing the transfer function. One also can note that in the context of dynamical systems that the matrices $\widetilde{\mathbf{V}}$ and $\widetilde{\mathbf{W}}$ defined in (2.8) are truncations of the observability and reachability matrices corresponding to the transfer function $\mathbf{H}(z)$.

4.2 Contour Integration with Rational Functions

Given this connection, we are motivated to develop a contour integration method using the Loewner realization technique. The first step seeks a suitable contour integral that will yield the data required in the Loewner framework. Recall that for any analytic function $f : \Omega \to \mathbb{C}$,

$$\frac{1}{2\pi i} \int_{\partial\Omega} f(z)\mathbf{T}(z)^{-1}\, dz = \mathbf{V} f(\mathbf{\Lambda})\mathbf{W}^*.$$

Let us define

$$f_\sigma(z) \coloneqq (\sigma - z)^{-1},$$

where $\sigma \in \mathbb{C}$. We can consider three possible cases for the location of σ compared to the search region Ω; when $\sigma \in \Omega$, $\sigma \in \partial\Omega$, or $\sigma \notin \Omega \cup \partial\Omega$.

Given that we have control over search region Ω, we can assume that we can avoid the second case; we provide contour integration results for the other two.

Theorem 4.2.1

Assume $\sigma \in \mathbb{C}$ and that σ is not an eigenvalue of $\mathbf{T}$. Using the Keldysh decomposition given in (4.1) and the definition of $\mathbf{H}$ given in (4.2), the following results hold.

1. *If $\sigma \notin \Omega \cup \partial\Omega$, then*
$$\frac{1}{2\pi i}\int_{\partial\Omega} f_\sigma(z)\mathbf{T}(z)^{-1}\, dz = \mathbf{H}(\sigma).$$
2. *If $\sigma \in \Omega$, then*
$$\frac{1}{2\pi i}\int_{\partial\Omega} f_\sigma(z)\mathbf{T}(z)^{-1}\, dz = -\mathbf{R}(\sigma).$$

Proof. If $\sigma \notin \Omega \cup \partial\Omega$, then f_σ is analytic on Ω. Therefore

$$\frac{1}{2\pi i}\int_{\partial\Omega} f_\sigma(z)\mathbf{T}(z)^{-1}\, dz = \mathbf{V} f_\sigma(\mathbf{\Lambda})\mathbf{W}^* = \mathbf{V}(\sigma\mathbf{I} - \mathbf{\Lambda})^{-1}\mathbf{W}^* = \mathbf{H}(\sigma).$$

Therefore we can evaluate $\mathbf{H}(z)$ at any point outside of Ω by computing a contour integral.

If $\sigma \in \Omega$, then f_σ is not analytic on Ω; we are in fact adding a pole to $\mathbf{T}(z)^{-1}$ located at σ. Then

$$\frac{1}{2\pi i}\int_{\partial\Omega} f_\sigma(z)\mathbf{T}(z)^{-1}\, dz = \underbrace{\frac{1}{2\pi i}\int_{\partial\Omega} f_\sigma(z)\mathbf{H}(z)\, dz}_{I_1} + \underbrace{\frac{1}{2\pi i}\int_{\partial\Omega} f_\sigma(z)\mathbf{R}(z)\, dz}_{I_2}.$$

To compute I_1, we use the calculus of residues, yielding

$$I_1 = \text{Res}(\sigma, f_\sigma(z)\mathbf{H}(z)) + \sum_{k=1}^{m} \text{Res}(\lambda_k, f_\sigma(z)\mathbf{H}(z)).$$

Since

$$\text{Res}(\lambda_k, f_\sigma(z)\mathbf{H}(z)) = \frac{\mathbf{v}_k \mathbf{w}_k^*}{\sigma - \lambda_k}, \text{ and } \text{Res}(\sigma, f_\sigma(z)\mathbf{H}(z)) = -\mathbf{H}(\sigma),$$

we conclude

$$I_1 = -\mathbf{H}(\sigma) + \sum_{k=1}^{m} \frac{\mathbf{v}_k \mathbf{w}_k^*}{\sigma - \lambda_k} = -\mathbf{H}(\sigma) + \mathbf{H}(\sigma) = 0.$$

By the Cauchy integral formula, we have

$$I_2 = -\mathbf{R}(\sigma).$$

Thus,

$$\frac{1}{2\pi i} \int_{\partial\Omega} f_\sigma(z)\mathbf{T}(z)^{-1}\, dz = -\mathbf{R}(\sigma).$$

□

We can then see that the Loewner matrices $\mathbb{L}$ and $\mathbb{L}_s$, and the matrices of left and right data $\mathbf{B}_r$ and $\mathbf{C}_r$, can be formed through contour integration. We provide results for the case where the interpolation points are located outside of search region Ω.

Theorem 4.2.2

Assume the sets of distinct interpolation points $\{\sigma_j\}_{j=1}^r$, $\{\mu_j\}_{j=1}^r$ *are not contained in* $\Omega \cup \partial\Omega$, *and are not eigenvalues of* $\mathbf{T}$. *Given interpolation directions* $\{\mathbf{r}_j\}_{j=1}^r$ *and* $\{\boldsymbol{\ell}_j\}_{j=1}^r$, *we obtain vectors of left and right data by*

$$\mathbf{b}_j = \mathbf{H}(\sigma_j)^* \boldsymbol{\ell}_j = -\frac{1}{2\pi i} \int_{\partial\Omega} \frac{1}{\overline{\sigma} - \overline{z}} \mathbf{T}(z)^{-*} \boldsymbol{\ell}_j\, dz,$$

and

$$\mathbf{c}_j = \mathbf{H}(\mu_j)\mathbf{r}_j = \frac{1}{2\pi i}\int_{\partial\Omega} \frac{1}{\sigma - z}\mathbf{T}(z)^{-1}\mathbf{r}_j \; dz$$

for $j = 1, \ldots r$. *Then the Loewner matrices can be obtained by*

$$(\mathbb{L})_{j,k} = \frac{\mathbf{b}_j^*\mathbf{r}_k - \boldsymbol{\ell}_j^*\mathbf{c}_k}{\sigma_j - \mu_k}, \quad (\mathbb{L}_s)_{j,k} = \frac{\sigma_j\mathbf{b}_j^*\mathbf{r}_k - \mu_k\boldsymbol{\ell}_j^*\mathbf{c}_k}{\sigma_j - \mu_k},$$

for $j, k = 1, \ldots, r$, *along with matrices of left and right data,*

$$\mathbf{C}_r = [\,\mathbf{c}_1, \ldots, \mathbf{c}_r\,], \quad \mathbf{B}_r = [\,\mathbf{b}_1, \ldots, \mathbf{b}_r\,]^*.$$

We now have a way to obtain the matrices required in Theorem 3.1.1. Assuming we set $r > m$, we can therefore identify the function $\mathbf{H}(z)$ exactly using the SVD truncation shown in Theorem 3.2.1. One can also note that we can in fact obtain the entries of the Loewner and shifted Loewner matrix using a single contour integral per entry.

Theorem 4.2.3

Assuming the sets of interpolation points $\{\sigma_j\}_{j=1}^r$, $\{\mu_j\}_{j=1}^r$ *are not contained in* $\Omega \cup \partial\Omega$, *then we obtain the entries of the Loewner matrices using the following contour integral formulas:*

$$(\mathbb{L})_{j,k} = -\frac{1}{2\pi i}\int_{\partial\Omega} f_{\sigma_j}(z)f_{\mu_k}(z)\boldsymbol{\ell}_j^*\mathbf{T}(z)^{-1}\mathbf{r}_k \; dz,$$

$$(\mathbb{L}_s)_{j,k} = -\frac{1}{2\pi i}\int_{\partial\Omega} zf_{\sigma_j}(z)f_{\mu_k}(z)\boldsymbol{\ell}_j^*\mathbf{T}(z)^{-1}\mathbf{r}_k \; dz,$$

where $\{\boldsymbol{\ell}_j\}_{j=1}^r, \{\mathbf{r}_j\}_{j=1}^r \subset \mathbb{C}^n$ *are the supplied interpolation directions.*

Proof. Given that $\sigma_j, \mu_k \notin \Omega \cup \partial\Omega$, the function $f_{\sigma_j}(z)f_{\mu_k}(z)$ is analytic in Ω. Therefore we

have

$$\frac{1}{2\pi i}\int_{\partial\Omega} f_{\sigma_j}(z)f_{\mu_k}(z)\mathbf{T}(z)^{-1}\,dz = \mathbf{V}f_{\sigma_j}(\mathbf{\Lambda})f_{\mu_k}(\mathbf{\Lambda})\mathbf{W}^* \\ = \mathbf{V}(\sigma_j\mathbf{I}-\mathbf{\Lambda})^{-1}(\mu_k\mathbf{I}-\mathbf{\Lambda})^{-1}\mathbf{W}^*.$$

Given the factorization of $\mathbb{L}$ into the generalized observability and reachability matrices, we have that

$$(\mathbb{L})_{j,k} = -\boldsymbol{\ell}_j^*\left(\frac{1}{2\pi i}\int_{\partial\Omega} f_{\sigma_j}(z)f_{\mu_k}(z)\mathbf{T}(z)^{-1}\,dz\right)\mathbf{r}_k,$$

and thus we compute the entries of Loewner matrix for $\mathbf{H}$ as

$$(\mathbb{L})_{j,k} = -\frac{1}{2\pi i}\int_{\partial\Omega} f_{\sigma_j}(z)f_{\mu_k}(z)\boldsymbol{\ell}_j^*\mathbf{T}(z)^{-1}\mathbf{r}_k\,dz.$$

Similarly for the entries of the shifted Loewner matrix, we can compute

$$\frac{1}{2\pi i}\int_{\partial\Omega} zf_{\sigma_j}(z)f_{\mu_k}(z)\mathbf{T}(z)^{-1}\,dz = \mathbf{V}f_{\sigma_j}(\mathbf{\Lambda})\mathbf{\Lambda}f_{\mu_k}(\mathbf{\Lambda})\mathbf{W}^* \\ = \mathbf{V}(\sigma_j\mathbf{I}-\mathbf{\Lambda})^{-1}\mathbf{\Lambda}(\mu_k\mathbf{I}-\mathbf{\Lambda})^{-1}\mathbf{W}^*$$

and so

$$(\mathbb{L}_s)_{j,k} = -\frac{1}{2\pi i}\int_{\partial\Omega} zf_{\sigma_j}(z)f_{\mu_k}(z)\boldsymbol{\ell}_j^*\mathbf{T}(z)^{-1}\mathbf{r}_k\,dz.$$

□

4.3 Obtaining Eigenvalue-Eigenvector Pairs

From the Loewner realization of $\mathbf{H}(z)$ given by

$$\mathbf{H}(z) \coloneqq \mathbf{C}_r\left(\mathbb{L}_s - z\mathbb{L}\right)^{-1}\mathbf{B}_r,$$

we can obtain the eigenvalues contained in Ω and their corresponding eigenvectors by diagonalizing the matrix pencil $(\mathbb{L}_s, \mathbb{L})$. If we assume the generalized eigenvalue decomposition for the Loewner pencil,

$$\mathbb{L}_s = \mathbb{L}\mathbf{X}\mathbf{\Lambda}\mathbf{X}^{-1} \tag{4.3}$$

then we obtain

$$\mathbf{H}(z) = \mathbf{C}_r\mathbf{X}(\mathbf{\Lambda} - z\mathbf{I})^{-1}\left(\mathbf{X}^{-1}\mathbb{L}^{-1}\mathbf{B}_r\right).$$

We can see that if the Loewner pencil $(\mathbb{L}_s, \mathbb{L})$ has eigen-triples $(\lambda_j, \mathbf{x}_j, \mathbf{y}_j)$ for $j = 1, \ldots m$, then $\mathbf{T}$ has eigen-triples $(\lambda_j, \mathbf{C}_r\mathbf{x}_j, \mathbf{B}_r^*\mathbf{y}_j)$. The process for the Loewner contour integration method is now summarized in Algorithm 5.

Algorithm 5 Loewner Contour Integration

input: left/right interpolation points $\{\sigma_j\}_{j=1}^r, \{\mu_j\}_{j=1}^r \subset \mathbb{C} - \Omega$, and left/right interpolation directions $\{\boldsymbol{\ell}_j\}_{j=1}^r, \{\mathbf{r}_j\}_{j=1}^r, \subset \mathbb{C}^n$:

1. Compute Loewner matrices $\mathbb{L}$ and $\mathbb{L}_s$, and matrices of left and right data $\mathbf{B}_r$ and $\mathbf{C}_r$ as detailed in Theorem 4.2.2.
2. Perform SVD truncation detailed in Theorem 3.2.1 to obtain reduced Loewner matrices $\widetilde{\mathbb{L}}$ and $\widetilde{\mathbb{L}}_s$, and matrices of left and right data $\widetilde{\mathbf{B}}_r$ and $\widetilde{\mathbf{C}}_r$.
3. Obtain eigen-triples of the Loewner pencil $(\widetilde{\mathbb{L}}, \widetilde{\mathbb{L}}_s)$ as $(\lambda_j, \mathbf{x}_j, \mathbf{y}_j)$ for $j = 1, \ldots, m$.
4. Obtain approximate eigen-triples of the nonlinear eigenvalue problem $(\lambda_j, \widetilde{\mathbf{C}}_r\mathbf{x}_j, \widetilde{\mathbf{B}}_r^*\mathbf{y}_j)$.

4.4 Numerical Experiments

We now show several numerical experiments for Algorithm 5 applied to the simple delay problem detailed in Example 1. A fundamental difference in implementation between the Loewner and Hankel contour integration methods is the requirement of choosing the location of the interpolation points $\{\sigma_j\}_{j=1}^r$ and $\{\mu_j\}_{j=1}^r$. We show results for several different location choices and describe the differences in the numerical results. We will compute the required contour integrals using the trapezoid rule detailed in Chapter 2. To draw comparisons with the results of the Hankel contour integration method, we use the same circular contour encompassing 200 eigenvalues. We also take $r = 300$, yielding Loewner matrices of size 300×300, as was the case with the Hankel matrices $\mathbf{H}_0$ and $\mathbf{H}_1$.

Figure 4.1 shows numerical results for several different interpolation point locations. The first column shows results when the interpolation points are taken on the circle of radius 6. In this case, we see less of a drop after 200 singular values compared to the other choices of interpolation points. We also see slower decay in the residual error of the eigenvalues. We see a substantial improvement in both how sharply the singular values drop and the residual error in the second column when the interpolation points are taken on the circle of radius 10. We see similar singular value and error results when taking the interpolation points to the left and right of the contours, shown the third and fourth columns.

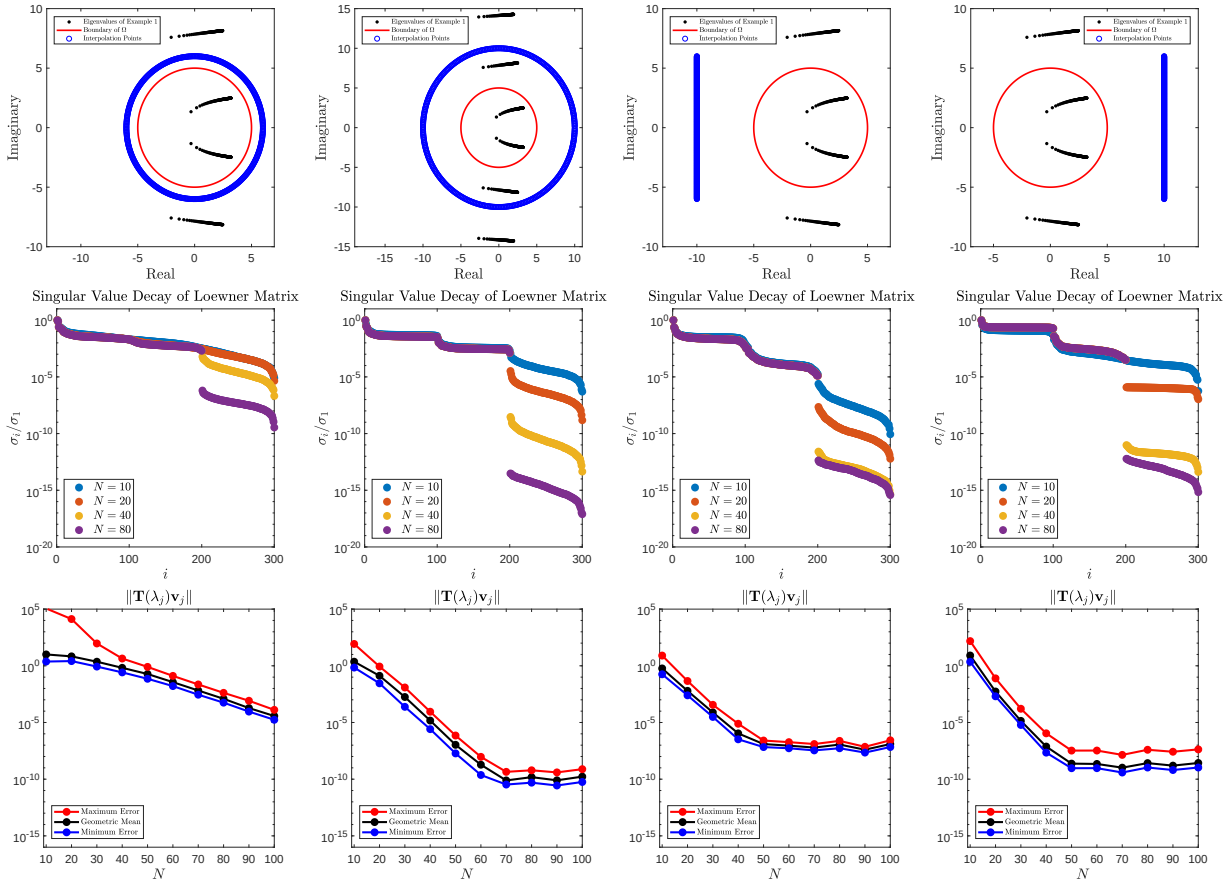

Figure 4.1: (Top) locations of interpolation points; (middle) Singular value decays of $\mathbb{L}$ for varying N; (bottom) Residual error decays

4.5 Analysis of Hankel and Loewner Methods Using Filter Functions

We now summarize the description of the Hankel contour integration methods by [44, 45] using the concept of filter functions, and find similar results for the Loewner contour inte-

gration method. This analysis provides a measure of how eigenvalues outside the region Ω are *filtered* out as the approximation of the integral improves. Recalling the definition

$$\mathbf{A}_p = \frac{1}{2\pi i} \int_{\partial\Omega} z^p \mathbf{T}(z)^{-1}\, dz,$$

we approximate $\mathbf{A}_p$ using a quadrature formula with N points on the contour ($z_j \in \partial\Omega$ for $j = 0, \ldots, N$) defined by

$$\mathbf{A}_p \approx \widetilde{\mathbf{A}}_p := \sum_{j=0}^{N-1} w_j z_j^p \mathbf{T}(z_j)^{-1}.$$

Using the Keldysh decomposition, where m is the number of eigenvalues contained in Ω, we have

$$\begin{aligned} \widetilde{\mathbf{A}}_p &= \sum_{j=0}^{N-1} w_j z_j^p \mathbf{H}(z_j) + \sum_{j=0}^{N-1} w_j z_j^p \mathbf{R}(z_j) \\ &= \sum_{k=1}^{m} \mathbf{v}_k \mathbf{w}_k^* \sum_{j=0}^{N-1} \frac{w_j z_j^p}{z_j - \lambda_k} + \sum_{j=0}^{N-1} w_j z_j^p \mathbf{R}(z_j). \end{aligned}$$

Then defining $b_p(z) := \sum_{j=0}^{N-1} \dfrac{w_j z_j^p}{z_j - z}$, we have

$$\widetilde{\mathbf{A}}_p = \sum_{k=1}^{m} \mathbf{v}_k \mathbf{w}_k^* b_p(\lambda_k) + \sum_{j=0}^{N-1} w_j z_j^p \mathbf{R}(z_j).$$

Using the trapezoidal rule where Ω is taken to be the unit disk $\Omega = \{z : |z| < 1\}$, points on the contour are taken to be $z_j = e^{2\pi i j/N}$, and the quadrature weights are taken to be $w_j = z_j/N$. In this case we obtain the simplification

$$b_0(z) = \frac{1}{N} \sum_{j=0}^{N-1} \frac{z_j}{z_j - z} = \frac{1}{1 - z^N},$$

and similarly

$$b_p(z) = \frac{z^p}{1 - z^N} = z^p b_0(z).$$

One can view $b_0(z)$ as the trapezoidal rule approximation of the unit step filter defined by the unit circle,

$$b_0(\lambda) \approx \frac{1}{2\pi i} \int_{\partial\Omega} \frac{1}{z-\lambda}\, dz = \begin{cases} 1, & |\lambda| < 1; \\ \\ 0, & |\lambda| > 1. \end{cases}$$

We can measure how well the eigenvalues outside the search region Ω are filtered out by how well $b_0(z)$ approximates this unit step function. We see that one desires the filter function to decay rapidly near eigenvalues exterior to Ω. We now provide similar analysis for the Loewner method where the rational function $f_\sigma(z) = (\sigma - z)^{-1}$ is placed in the contour integral. Rather than approximating the term $\mathbf{A}_p$, we approximate an arbitrary vector of right data, and obtain

$$\frac{1}{2\pi i} \int_{\partial\Omega} \frac{1}{\sigma - z} \mathbf{T}(z)^{-1}\mathbf{r}\, dz \approx \sum_{k=1}^{m} \mathbf{v}_k \mathbf{w}_k^* \mathbf{r} \sum_{j=0}^{N-1} \frac{w_j}{(z_j - \lambda_k)(\sigma - z_j)} + \sum_{j=0}^{N-1} \frac{w_j \mathbf{R}(z_j)\mathbf{r}}{\sigma - z_j}$$

where $\mathbf{r}$ is an interpolation direction and σ is an interpolation point. Assuming $\sigma \neq \lambda_k$ for $k = 1, \ldots, m$, then

$$\frac{w_j}{(z_j - \lambda_k)(\sigma - z_j)} = \frac{w_j}{\sigma - \lambda_k} \left(\frac{1}{z_j - \lambda_k} - \frac{1}{z_j - \sigma} \right).$$

Using the trapezoidal rule with $w_j = z_j/N$ and $z_j = e^{2\pi i j/N}$, we then have the filter function

$$b_\sigma(\lambda) = \sum_{j=0}^{N-1} \frac{w_j}{(z_j - \lambda)(\sigma - z_j)} = \left(\frac{1}{\sigma - \lambda} \right) \frac{1}{N} \sum_{j=0}^{N-1} \left(\frac{z_j}{z_j - \lambda} - \frac{z_j}{z_j - \sigma} \right).$$

Given that

$$\frac{1}{N} \sum_{j=0}^{N-1} \frac{z_j}{z_j - z} = \frac{1}{1 - z^N}$$

we have

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

Note that given that we assume $\sigma \notin \Omega$, (in this case meaning $|\sigma| > 1$) we have

$$b_\sigma(\lambda) \approx \frac{1}{2\pi i}\int_{\partial\Omega}\left(\frac{1}{z-\lambda}\right)\left(\frac{1}{\sigma - z}\right)\, dz = \begin{cases} \frac{1}{\sigma-\lambda}, & |\lambda| < 1; \\ \\ 0, & |\lambda| > 1. \end{cases}$$

Figure 4.2 shows cross sections of the filter functions corresponding to the Hankel and Loewner contour integration methods for $\sigma = -1.1$, -2, and -4. The number of points used in the trapezoid rule approximation is taken to be $N = 25$ in each case. We can see that $b_0(z)$ is approximately 1 inside the unit interval, where as $b_\sigma(z)$ approximates the function $(\sigma - z)^{-1}$ inside the unit interval. Notably, we see that $b_\sigma(z)$ decays to zero for $z > 1$ more rapidly then $b_0(z)$ for these choices of σ. This suggests that when Ω is taken as a disk, that the interpolation points should be set on the opposite side of the closest exterior eigenvalue.

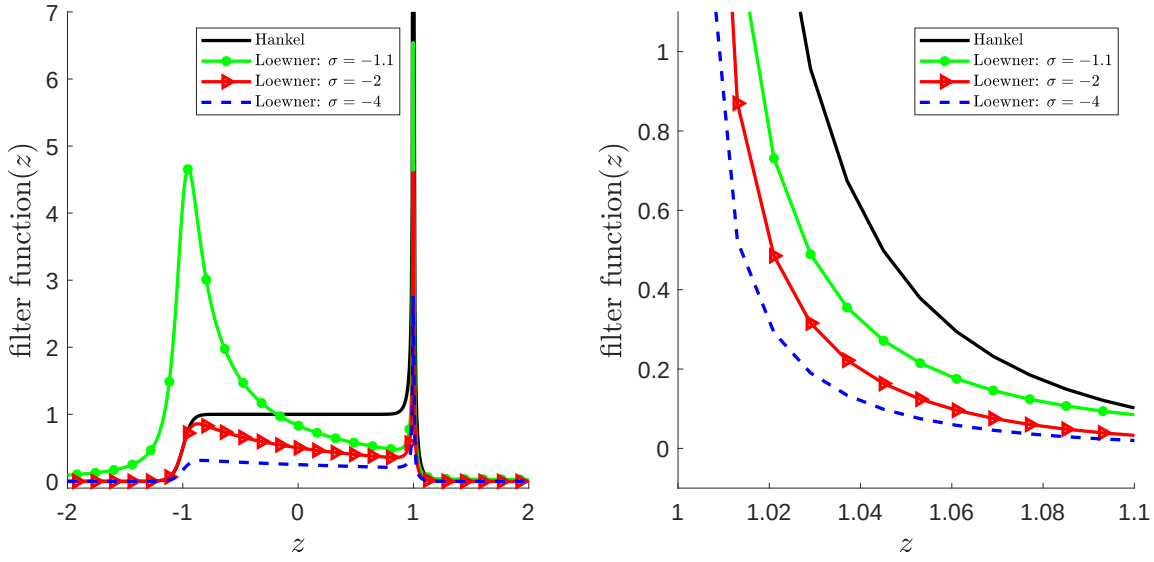

Figure 4.2: Comparing filter functions from the Hankel and Loewner contour integration methods

A preliminary test of this concept is included below, where Ω is defined to include six of the right most eigenvalues of a particular branch of solutions, with the seventh right most eigenvalue lying close to the boundary of Ω. In this situation one would desire the filter function to decay rapidly near the seventh eigenvalue. The result shown in Figure 4.2 then suggests taking the interpolation points to the right of the contour. In this case, we take $r = 10$ yielding Loewner matrices of size 10×10. We again compute the contour integrals using the trapezoid rule. Figure 4.3 shows results for several choices of interpolation point locations. Although we see similar singular value decays in each case, the residual error is worst when we take the interpolation points on a circle surrounding Ω, compared to taking the interpolation points on an arch that is to the right of Ω.

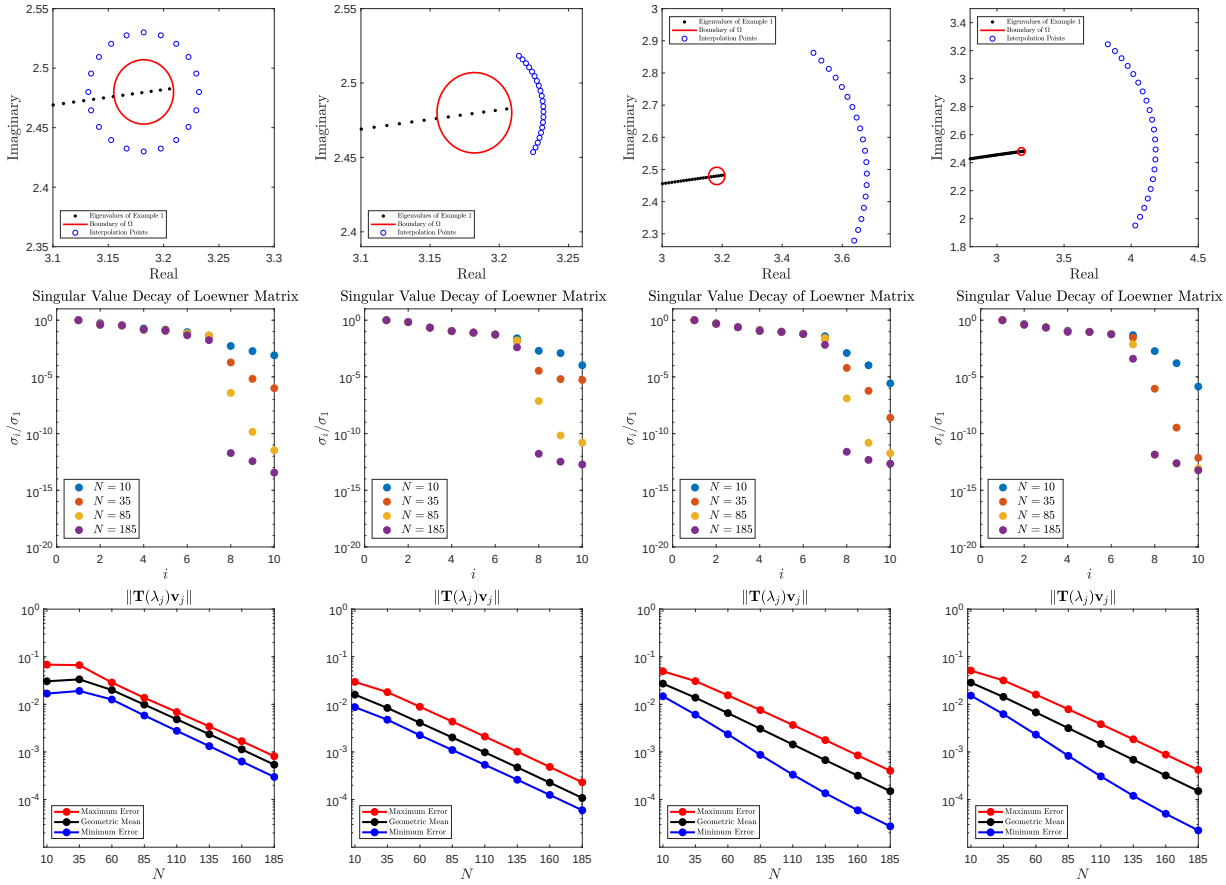

Figure 4.3: (Top) locations of interpolation points; (middle) Singular value decays of $\mathbb{L}$ for varying N; (bottom) Residual error decays

Table 4.1 summarizes the residual errors of the approximate eigenvalues when $N = 185$. We see that the minimum error in the eigenvalues is smallest for column 4, when taking the interpolation points to the right of Ω and farthest away, while the geometric mean and the maximum error is smallest for column 2 when they are to the right of Ω and closer.

Table 4.1: Residual error results for $N = 185$

Location of interpolation points	Geometric mean residual error	Maximum residual error	Minimum residual error
Column 1	3.3979×10^{-4}	4.9542×10^{-4}	2.1282×10^{-4}
Column 2	1.0815×10^{-4}	2.3047×10^{-4}	5.9679×10^{-5}
Column 3	1.5033×10^{-4}	4.0280×10^{-4}	2.7478×10^{-5}
Column 4	1.5110×10^{-4}	4.1898×10^{-4}	2.2517×10^{-5}

Chapter 5

Eigenvalue approximation via Rational Interpolation

We now examine the use of rational interpolation to approximate a set of eigenvalues for a nonlinear eigenvalue problem. In the previous chapter pertaining to contour integration methods, we saw that the Keldysh decomposition,

$$\mathbf{T}(z)^{-1} = \mathbf{V}(z\mathbf{I} - \mathbf{\Lambda})^{-1}\mathbf{W}^* + \mathbf{R}(z),$$

exposes an underlying linear transfer function $\mathbf{H}(z) = \mathbf{V}(z\mathbf{I} - \mathbf{\Lambda})^{-1}\mathbf{W}^*$. We used contour integration to remove contributions from the residual nonlinear term $\mathbf{R}(z)$ and rational interpolation to identify $\mathbf{H}(z)$. Now, we view $\mathbf{T}(z)^{-1}$ as a general nonlinear transfer function that we approximate with a rational interpolant. One key difference between this method and rational linearization methods discussed in Chapter 2, is how we obtain the approximate

eigenvalues. In linearization methods, one forms the rational function $\mathbf{Q}$ such that $\mathbf{Q} \approx \mathbf{T}$ and uses the eigenvalues of $\mathbf{Q}$ as approximations of the eigenvalues of $\mathbf{T}$. The dimension of the resulting approximate problem is typically larger than the dimension of the nonlinear problem. Defining

$$\mathbf{F}(z) \coloneqq \mathbf{T}(z)^{-1},$$

we will form $\mathbf{G}$ such that $\mathbf{G} \approx \mathbf{F}$, and we use the poles of $\mathbf{G}$ to approximate the poles of $\mathbf{F}$, equivalently the eigenvalues of $\mathbf{T}$. We use the Loewner methodology, where $\mathbf{G}(z)$ is formed as a rational interpolant as detailed in Chapter 3. We find this framework attractive due to its wide development in model reduction and because the resulting linear problem is of small dimension.

5.1 Eigenvalue Approximation by Poles

The matrix valued function $\mathbf{T}(z)^{-1}$ has poles at the eigenvalues of $\mathbf{T}$. The method proposed here approximates $\mathbf{T}(z)^{-1}$ using a rational function $\mathbf{G}(z)$,

$$\mathbf{G}(z) \coloneqq \mathbf{C}_r \left(\mathbb{L}_s - z\mathbb{L}\right)^{-1} \mathbf{B}_r \approx \mathbf{T}(z)^{-1}$$

where the Loewner matrices $\mathbb{L}$ and $\mathbb{L}_s$, and matrices of left and right data $\mathbf{B}_r$ and $\mathbf{C}_r$ are defined in Theorem 3.1.1, where the function $\mathbf{H}(z)$ is replace by $\mathbf{T}(z)^{-1}$. We then approximate the poles of $\mathbf{T}(z)^{-1}$ with the poles of $\mathbf{G}(z)$, obtaining approximate eigenvalues for $\mathbf{T}$. Approximate eigenvectors are obtain by diagonalizing the Loewner pencil in the same fashion detailed in Section 4.3. We now summarize this method in Algorithm 6.

Algorithm 6 Rational interpolation: Approximation by poles

input: left/right interpolation points $\{\sigma_j\}_{j=1}^r, \{\mu_j\}_{j=1}^r \subset \mathbb{C}$, and left/right interpolation directions $\{\boldsymbol{\ell}_j\}_{j=1}^r, \{\mathbf{r}_j\}_{j=1}^r, \subset \mathbb{C}^n$:

1. Compute Loewner matrices $\mathbb{L}$ and $\mathbb{L}_s$, and matrices of left and right data $\mathbf{B}_r$ and $\mathbf{C}_r$ as detailed in Theorem 3.1.1 using $\mathbf{T}(z)^{-1}$.
2. Obtain eigen-triples of the Loewner pencil $(\mathbb{L}_s, \mathbb{L})$ as $(\lambda_j, \mathbf{x}_j, \mathbf{y}_j)$ for $j = 1, \ldots, m$
3. Obtain approximate eigen-triples of the nonlinear eigenvalue problem $(\lambda_j, \mathbf{C}_r\mathbf{x}_j, \mathbf{B}_r^*\mathbf{y}_j)$

Note that the primary computational cost comes from forming the Loewner and data matrices, which requires $2r$ factorizations of $\mathbf{T}(z)$ in the Lagrange case and r factorizations in the Hermite case, yielding a computational cost of $\mathcal{O}(n^3 r)$ when we assume that $\mathbf{T}(z)$ is generally dense. Once these are formed, the resulting generalized linear eigenvalue problem is of size r, which we expect to be significantly less than n. One can also note that one obtains r approximate eigenvalues, given the Loewner pencil will be of dimension r. We will assume that we have access to $\mathbf{T}'(z)$ and therefore we choose to work with the Hermite case. Given that we desire our rational approximations to be accurate near the poles of $\mathbf{T}(z)^{-1}$, we assume that the interpolation points are distributed around a particular search region or that they are rough approximations to eigenvalues of interest. This leaves us with the task of choosing interpolation directions. Later we will show that the eigenvectors of the approximated eigenvalues are very successful interpolation directions.

5.2 Numerical Experiments

We now provide numerical results for Example 1, the simple delay case detailed in Chapter 1. The following figures show results of Algorithm 6 using four interpolation points close to distinct eigenvalues (the same interpolation points used in Chapter 2 when discussing polynomial linearization). In Figure 5.1, we show the resulting approximate eigenvalues using Algorithm 6, for three different choices for interpolation directions: random directions, the singular vectors corresponding to the smallest singular values of $\mathbf{T}(\sigma_j)$ for $j = 1, \ldots, r$ and finally the exact eigenvectors corresponding to the nearest eigenvalue. We see that using random directions yields the poorest approximations. Using singular vectors we achieve approximations of one eigenvalue with eigenvalue error less then 10^{-3}, and see that using the exact eigenvectors yields the best results, approximating four eigenvalues with eigenvalue error less then 10^{-3}.

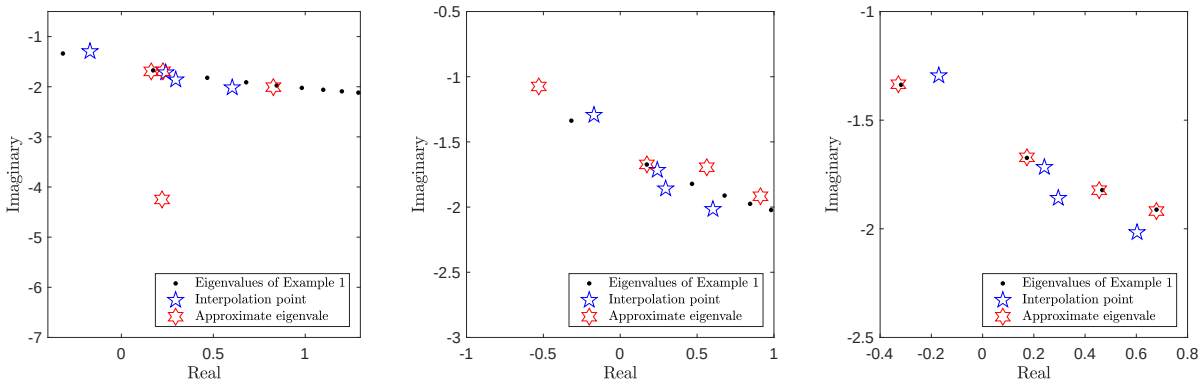

Figure 5.1: Results of Algorithm 6: (left) Using random interpolation directions; (center) singular vectors; (right) exact eigenvectors.

5.3 Iterative Rational Interpolation

We now examine the use of rational interpolation to approximate a set of eigenvalues for a nonlinear eigenvalue problem iteratively. The general idea of the iterative methods built here is to iteratively form rational interpolants of $\mathbf{T}(z)^{-1}$, using the approximate eigenvalues from each step as new interpolation points. We will also note that this method is closely related to the *iterative rational Krylov algorithm* using transfer function evaluations (TF-IRKA) [12] and the *dominant pole algorithm* [36]. First, we focus on the case where we limit the method to one interpolation point, meaning we take $r = 1$ and approximate one eigenvalue of $\mathbf{T}$. We will see that this method is theoretically equivalently to Newton's method. Given that the interpolant depends on the interpolation directions as well, we also must decide on how these directions can be updated.

5.3.1 Single Interpolation Point

We first describe the most basic case for this method, where we keep the interpolation directions constant. We will see that this leads to the implicit use of Newton's method to find the poles of a particular scalar function.

Proposition 5.3.1

The iterative application of Algorithm 6, setting $r = 1$, updating the interpolation point to be the approximate eigenvalue yielded from the previous step, and using constant interpolation

directions, $\boldsymbol{\ell}, \mathbf{r} \in \mathbb{C}^n$ *is equivalent to the application of Newton's method to the scalar function*

$$g(z) \coloneqq \frac{1}{\boldsymbol{\ell}^* \mathbf{T}(z)^{-1} \mathbf{r}}.$$

Proof. Let us define $\mathbf{F}(z) \coloneqq \mathbf{T}(z)^{-1}$ to simplify the following calculation. In the case where we interpolate $\mathbf{F}(z)$ at one interpolation point, $z = z_0$ along right and left directions $\mathbf{r}$ and $\boldsymbol{\ell}$, the Loewner matrices reduce to scalars, yielding

$$\mathbb{L} = \boldsymbol{\ell}^* \mathbf{F}'(z_0) \mathbf{r},$$

and

$$\mathbb{L}_s = \boldsymbol{\ell}^* \left(z_0 \mathbf{F}'(z_0) + \mathbf{F}(z_0) \right) \mathbf{r} = z_0 \mathbb{L} + \boldsymbol{\ell}^* \mathbf{F}(z_0) \mathbf{r}.$$

The generalized eigenvalue problem yielding the poles of this interpolant reduces to the linear scalar equation,

$$\mathbb{L}_s - z_1 \mathbb{L} = 0,$$

where z_1 denotes the single pole. Assuming $\mathbb{L} \neq 0$, we see

$$z_1 = \frac{\mathbb{L}_s}{\mathbb{L}} = \frac{\boldsymbol{\ell}^* \left(z_0 \mathbf{F}'(z_0) + \mathbf{F}(z_0) \right) \mathbf{r}}{\boldsymbol{\ell}^* \mathbf{F}'(z_0) \mathbf{r}} = z_0 + \frac{\boldsymbol{\ell}^* \mathbf{F}(z_0) \mathbf{r}}{\boldsymbol{\ell}^* \mathbf{F}'(z_0) \mathbf{r}} = z_0 + \frac{f(z_0)}{f'(z_0)},$$

where we have defined the scalar function

$$f(z) \coloneqq \boldsymbol{\ell}^* \mathbf{F}(z) \mathbf{r}. \tag{5.1}$$

Expanding this to an iterative scheme, we obtain a recursive formula for a sequence of approximate eigenvalues:

$$z_{i+1} = z_i + \frac{f(z_i)}{f'(z_i)}. \tag{5.2}$$

This is in fact Newton's method applied to $g(z) = 1/f(z)$, noting that poles of f will be roots of g. □

To compute $f'(z)$, we require an identity for the derivative of the inverse of a matrix-valued function:

$$\frac{d}{dz}(\mathbf{T}(z)^{-1}) = -\mathbf{T}(z)^{-1}\mathbf{T}'(z)\mathbf{T}(z)^{-1}.$$

We now summarize this method in Algorithm 7.

Algorithm 7 Iterated Rational Interpolation with One Point (IRI-1)

input: initial interpolation point $z_0 \in \mathbb{C}$, and left/right interpolation directions $\boldsymbol{\ell}, \mathbf{r} \in \mathbb{C}^n$:

for $k = 0, 1, \ldots$ until converged **do**

1. Compute $f_k = \boldsymbol{\ell}^*\mathbf{T}(z_k)^{-1}\mathbf{r}$ and $f_k' = -\boldsymbol{\ell}^*\mathbf{T}(z_k)^{-1}\mathbf{T}'(z_k)\mathbf{T}(z_k)^{-1}\mathbf{r}$
2. Set $z_{k+1} = z_k + \dfrac{f_k}{f_k'}$.

end for

In the case where $f(z) = \mathbf{c}^*(\mathbf{E} - z\mathbf{A})^{-1}\mathbf{b}$, i.e., the transfer function of the single input - single output linear dynamical system,

$$\mathbf{E}\mathbf{x}'(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t)$$

$$y(t) = \mathbf{c}^*\mathbf{x}(t),$$

Algorithm 7 is specifically known as the *dominant pole algorithm* (DPA) [36]. DPA looks to locate the *dominant pole* of a linear dynamical system, meaning the pole with the largest

magnitude corresponding residue in a pole-residue decomposition. This connection indicates that we can influence convergence to particular eigenvalues based on our choice of interpolation directions. This also means we can find poor choices of interpolation directions for a given problem. Consider

$$\mathbf{T}(z) = \begin{bmatrix} d(z) & 0 \\ 0 & -d(z) \end{bmatrix}, \quad d : \mathbb{C} \to \mathbb{C}, \quad \mathbf{r} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \boldsymbol{\ell} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \tag{5.3}$$

Here we see $\mathbf{T}$ is diagonal and the eigenvalues are given by the roots of the scalar function $d(z)$. But given this choice of directions, the scalar function f is identically 0, and we lose all information regarding the roots of d, and thus the eigenvalues of $\mathbf{T}$. We desire interpolation directions that preserve the poles of the underlying function. We now show that this depends on the eigenvectors of $\mathbf{T}$ and build intuition for suitable choices for interpolation directions.

5.3.2 Updating Interpolation Directions

To understand the convergence of the recurrence relationship defined in (5.2), we require information relating the poles of the scalar function f to the poles of $\mathbf{T}(z)^{-1}$. Recall the Keldysh decomposition stated in Theorem 2.3.1, and assume that $\mathbf{T}$ has m semi-simple eigenvalues denoted $\lambda_1, \ldots, \lambda_m$ in the domain $\Omega \subseteq \mathbb{C}$, counting multiplicities. Then we can express $\mathbf{T}(z)^{-1}$ as

$$\mathbf{T}(z)^{-1} = \mathbf{V}(z\mathbf{I} - \boldsymbol{\Lambda})^{-1}\mathbf{W}^* + \mathbf{R}(z) = \sum_{j=1}^{m} \frac{\mathbf{v}_j \mathbf{w}_j^*}{z - \lambda_j} + \mathbf{R}(z),$$

where $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_m]$, $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_m]$ are the right and left eigenvectors satisfying the normalization condition $\mathbf{w}_j^* \mathbf{T}'(\lambda_j)\mathbf{v}_j = 1$, $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_m)$, and $\mathbf{R} : \mathbb{C} \to \mathbb{C}^{n \times n}$ is

analytic in Ω. Thus we can express the scalar function f as

$$f(z) = \boldsymbol{\ell}^* \mathbf{T}(z)^{-1}\mathbf{r} = \sum_{j=1}^{m} \frac{(\boldsymbol{\ell}^* \mathbf{v}_j)(\mathbf{w}_j^* \mathbf{r})}{z - \lambda_j} + \boldsymbol{\ell}^* \mathbf{R}(z)\mathbf{r}.$$

We see explicitly that the only possible poles of f contained in Ω are the eigenvalues $\lambda_1, \ldots, \lambda_m$, and that whether they are preserved or omitted from f depends on the inner products between the eigenvectors and interpolation directions. Suppose we start with $z_0 \approx \lambda_k$ where $\lambda_k \in \{\lambda_1, \ldots, \lambda_m\}$. There are several limiting cases. Let us assume that $m < n$, and that the columns of $\mathbf{V}$ and $\mathbf{W}$ are linearly independent. Then ideally, we would pick $\boldsymbol{\ell}$ and $\mathbf{r}$ such that

$$\boldsymbol{\ell}^* \mathbf{v}_j = \mathbf{w}_j^* \mathbf{r} = \begin{cases} 1, & \text{if } j = k; \\ 0, & \text{if } j \neq k, \end{cases} \tag{5.4}$$

and obtain

$$f(z) = \frac{1}{z - \lambda_k} + \boldsymbol{\ell}^* \mathbf{R}(z)\mathbf{r}.$$

We see that this choice of interpolation directions isolates λ_k as the only pole of f contained in Ω. Here we are guaranteed that if the iteration defined in (5.2) converges to a point in Ω, then that point is λ_k. However, note that the ability to enforce the condition (5.4) requires that none of the eigenvalues in Ω share eigenvectors, which we have seen in Chapter 1 is not always true for nonlinear eigenvalue problems. Assuming the case where each of the eigenvalues of $\mathbf{T}$ contained in Ω is simple and shares the same eigenvectors, we obtain $\boldsymbol{\ell}^* \mathbf{v}_k = \boldsymbol{\ell}^* \mathbf{v}_j$ and $\mathbf{w}_k^* \mathbf{r} = \mathbf{w}_j^* \mathbf{r}$ for $j, k = 1, \ldots, m$. In this case, the choice of interpolation directions cannot omit any pole of $\mathbf{T}(z)^{-1}$ in Ω without omitting every pole of $\mathbf{T}(z)^{-1}$ in Ω.

5.3.3 Analysis Using a Spectral Decomposition

We saw above that the choice of interpolation directions influences the eigenvalue to which the iteration can converge. But given that we typically do not have information regarding the function $\mathbf{R}$, it is difficult to understand how this Newton iteration will behave for the scalar function f. We now show that for specific problems, we can gain greater insight. Let us assume $\mathbf{T}$ has the spectral decomposition

$$\mathbf{T}(z) = \mathbf{W}^{-*}\mathbf{D}(z)\mathbf{V}^{-1} \tag{5.5}$$

where

$$\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_n], \quad \text{and} \qquad \mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_n] \in \mathbb{C}^{n\times n}$$

are right and left eigenvectors of $\mathbf{T}$, and

$$\mathbf{D}(z) = \operatorname{diag}(d_1(z), \ldots, d_n(z)),$$

where $d_j : \mathbb{C} \to \mathbb{C}$, for $j = 1, \ldots, n$. Also we denote

$$\mathbf{V}^{-1} = \begin{bmatrix} \tilde{\mathbf{v}}_1^* \\ \vdots \\ \tilde{\mathbf{v}}_n^* \end{bmatrix}, \quad \text{and} \quad \mathbf{W}^{-*} = [\tilde{\mathbf{w}}_1, \ldots, \tilde{\mathbf{w}}_n] \in \mathbb{C}^{n\times n},$$

yielding

$$\mathbf{T}(z) = \sum_{j=1}^{n} \widetilde{\mathbf{w}}_j \widetilde{\mathbf{v}}_j^* d_j(z).$$

This decomposition exposes $\mathbf{v}_j$ and $\mathbf{w}_j$ as eigenvectors for all eigenvalues corresponding to the roots of the scalar functions d_j. This form is restrictive, as we saw in Chapter 1 that this

is not always possible even for the quadratic eigenvalue problem unless each of the matrices are simultaneously diagonalizable. However, consider problems of the form

$$\mathbf{T}(z) = g_1(z)\mathbf{A}_1 + g_2(z)\mathbf{A}_2, \tag{5.6}$$

where $\mathbf{A}_1, \mathbf{A}_2 \in \mathbb{C}^{n\times n}$, $g_1, g_2 : \mathbb{C} \to \mathbb{C}$, and $\mathbf{A}_2$ is nonsingular. Note that the basic delay problem of Example 1 detailed in Chapter 1 comes in this form with $g_1(z) = z$ and $g_2(z) = -e^{\tau z}$. If a generalized eigenvalue decomposition between $\mathbf{A}_1$ and $\mathbf{A}_2$ exists, i.e.,

$$\mathbf{A}_1 = \mathbf{A}_2\mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1},$$

then one can write equation (5.6) as

$$\begin{aligned}
\mathbf{T}(z) &= g_1(z)\mathbf{A}_1 + g_2(z)\mathbf{A}_2 \\
&= g_1(z)\mathbf{A}_2\mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1} + g_2(z)\mathbf{A}_2 \\
&= \mathbf{A}_2\left(g_1(z)\mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1} + g_2(z)\mathbf{I}\right) \\
&= \mathbf{A}_2\mathbf{V}\left(g_1(z)\mathbf{\Lambda} + g_2(z)\mathbf{I}\right)\mathbf{V}^{-1} \\
&= \mathbf{W}^{-*}\mathbf{D}(z)\mathbf{V}^{-1},
\end{aligned}$$

where $\mathbf{D}(z) = g_1(z)\mathbf{\Lambda} + g_2(z)\mathbf{I}$, and $\mathbf{W}^{-*} = \mathbf{A}_2\mathbf{V}$. Therefore, the following results do apply to several common problems. Given the form (5.5), we have an explicit formula for the inverse of $\mathbf{T}(z)$:

$$\mathbf{T}(z)^{-1} = \mathbf{V}\mathbf{D}(z)^{-1}\mathbf{W}^* = \sum_{j=1}^{n} \frac{\mathbf{v}_j\mathbf{w}_j^*}{d_j(z)}. \tag{5.7}$$

Note that this is similar to the Keldysh decomposition in that the eigenvalues and eigenvectors are exposed. However, in this form the repetition of eigenvectors for distinct eigenvalues

is captured as well. Then the scalar function f defined in (5.1) can be expressed as

$$f(z) = \boldsymbol{\ell}^* \mathbf{T}(z)^{-1} \mathbf{r} = \sum_{j=1}^{n} \frac{(\boldsymbol{\ell}^* \mathbf{v}_j)(\mathbf{w}_j^* \mathbf{r})}{d_j(z)}.$$

By writing the left and right interpolation directions in the convenient bases $\{\widetilde{\mathbf{w}}_1, \ldots, \widetilde{\mathbf{w}}_n\}$ and $\{\widetilde{\mathbf{v}}_1, \ldots, \widetilde{\mathbf{v}}_n\}$, we obtain

$$\mathbf{r} = \sum_{k=1}^{n} a_k \widetilde{\mathbf{w}}_k, \quad \text{and} \quad \boldsymbol{\ell}^* = \sum_{k=1}^{n} b_k \widetilde{\mathbf{v}}_k^*,$$

where $\{a_j\}_{j=1}^n$ and $\{b_j\}_{j=1}^n$ denote basis coefficients. Then for $j = 1, \ldots, n$, we have $\boldsymbol{\ell}^* \mathbf{v}_j = b_j$ and $\mathbf{w}_j^* \mathbf{r} = a_j$. Defining $c_j = a_j b_j$, we can express the scalar function f as a sum of the reciprocals of the functions d_j:

$$f(z) = \sum_{j=1}^{n} \frac{c_j}{d_j(z)}.$$

Therefore the iteration given in (5.2) implicitly applies Newton's method to find the poles of f, which are the roots of the functions $d_j(z)$. This also explains how to avoid the issue exposed in example (5.3), where f was identically zero for particular interpolation directions. In that case, we had $d_1(z) = -d_2(z) = d(z)$ and $c_1 = c_2$, yielding $f(z) = 0$ for all $z \in \mathbb{C}$. If we define

$$\mathbf{r} = \begin{bmatrix} r_1 \\ \\ r_2 \end{bmatrix}, \quad \boldsymbol{\ell} = \begin{bmatrix} \ell_1 \\ \\ \ell_2 \end{bmatrix},$$

then we have

$$f(z) = \frac{r_1 \ell_1 - r_2 \ell_2}{d(z)},$$

and so long as $r_1 \ell_1 \neq r_2 \ell_2$, f will still have poles at the roots of d.

As before, if given full knowledge of the spectral decomposition of $\mathbf{T}(z)$, we would take $\mathbf{r} = \widetilde{\mathbf{w}}_j$ and $\boldsymbol{\ell}^* = \widetilde{\mathbf{v}}_j^*$ for some integer $1 \leq j \leq n$, and we would obtain

$$f(z) = \frac{1}{d_j(z)}.$$

In this case, it is only possible to converge to poles corresponding to the roots of d_j. Moreover, plugging in $f'(z) = -\frac{d_j'(z)}{d_j(z)^2}$ into (5.2), we obtain

$$z_{i+1} = z_i + \frac{f(z_i)}{f'(z_i)} = z_i + \frac{\frac{1}{d_j(z_i)}}{-\frac{d_j'(z_i)}{d_j(z_i)^2}} = z_i - \frac{d_j(z_i)}{d_j'(z_i)}.$$

From this we see this method implicitly applies Newton's method to the function d_j.

One typically does not have full knowledge of this decomposition of $\mathbf{T}(z)$. Let us assume our interpolation directions are not exactly $\mathbf{r} = \widetilde{\mathbf{w}}_j$ and $\boldsymbol{\ell}^* = \widetilde{\mathbf{v}}_j^*$, but that they have perturbations in the direction of other eigenvectors, i.e.,

$$\mathbf{r} = \widetilde{\mathbf{w}}_j + \epsilon_1 \widetilde{\mathbf{w}}_q, \quad \text{and} \quad \boldsymbol{\ell}^* = \widetilde{\mathbf{v}}_j^* + \epsilon_2 \widetilde{\mathbf{v}}_q^*$$

for some integers $j, q \in \{1, 2, \ldots, n\}$. Then we obtain

$$f(z) = \frac{1}{d_j(z)} + \frac{\epsilon}{d_q(z)}, \tag{5.8}$$

where we define $\epsilon = \epsilon_1 \epsilon_2$. In this setting, we may hope to approximate the eigenvalues corresponding to the roots of d_j, but iteration (5.2) can in fact converge to poles corresponding to the roots of d_q. One can hope to expect the iteration would converge to a solution corresponding to $d_j(z) = 0$ if ϵ is small. In the linear case, this is shown explicitly in [36]. In our case, we visualize the basins of attraction to measure the attraction of different eigenvalues.

We use the simple delay problem given in Example 1 in Chapter 1. For this case, we can explicitly see the forms of d_k to be

$$d_k(z) = z - e^{-\tau z}\lambda_k,$$

where $\lambda_k = -k \in \sigma(\mathbf{A})$. Figure 5.2 shows the poles of $f(z)$ defined by (5.8) when $\epsilon \neq 0$, setting $j = 1$ and $q = 2$ and thus $\lambda_j = -1$ and $\lambda_q = -2$.

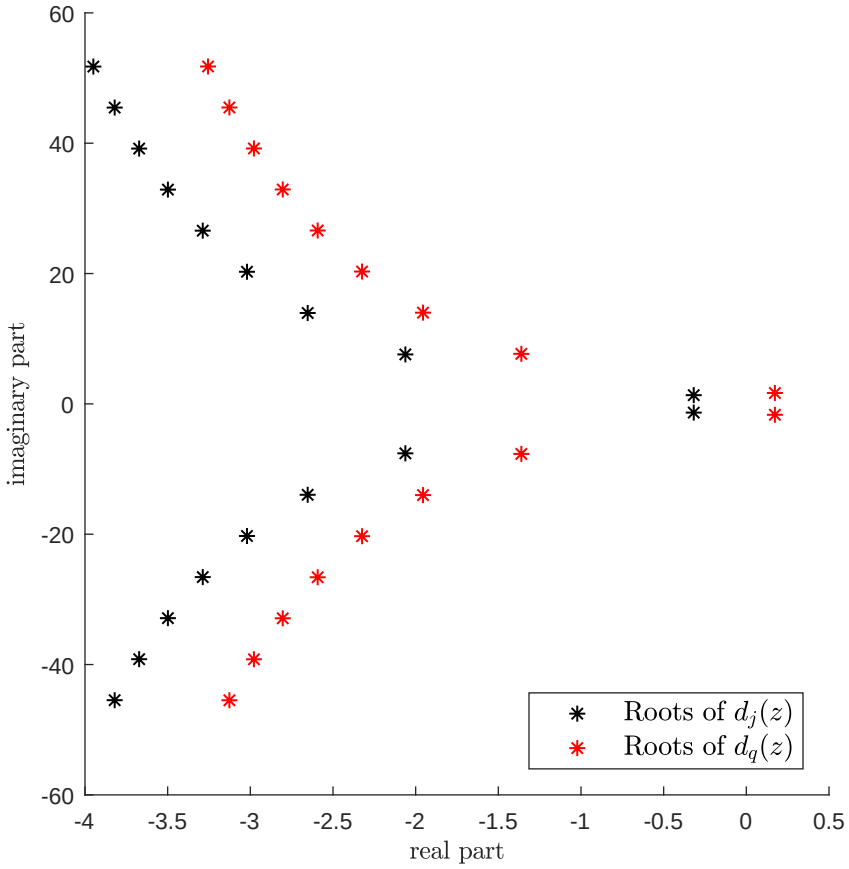

Figure 5.2: Poles of $f(z)$ defined in Equation (5.8).

Figure 5.3 shows how the basins of attraction depend on the magnitude of ϵ applying Algorithm 7 with constant interpolation directions of the form

$$\mathbf{r} = \widetilde{\mathbf{w}}_j + \epsilon_1 \widetilde{\mathbf{w}}_q, \quad \text{and} \quad \boldsymbol{\ell}^* = \widetilde{\mathbf{v}}_j^* + \epsilon_2 \widetilde{\mathbf{v}}_q^*.$$

As ϵ grows the basins of convergence for the roots of d_q begin to appear.

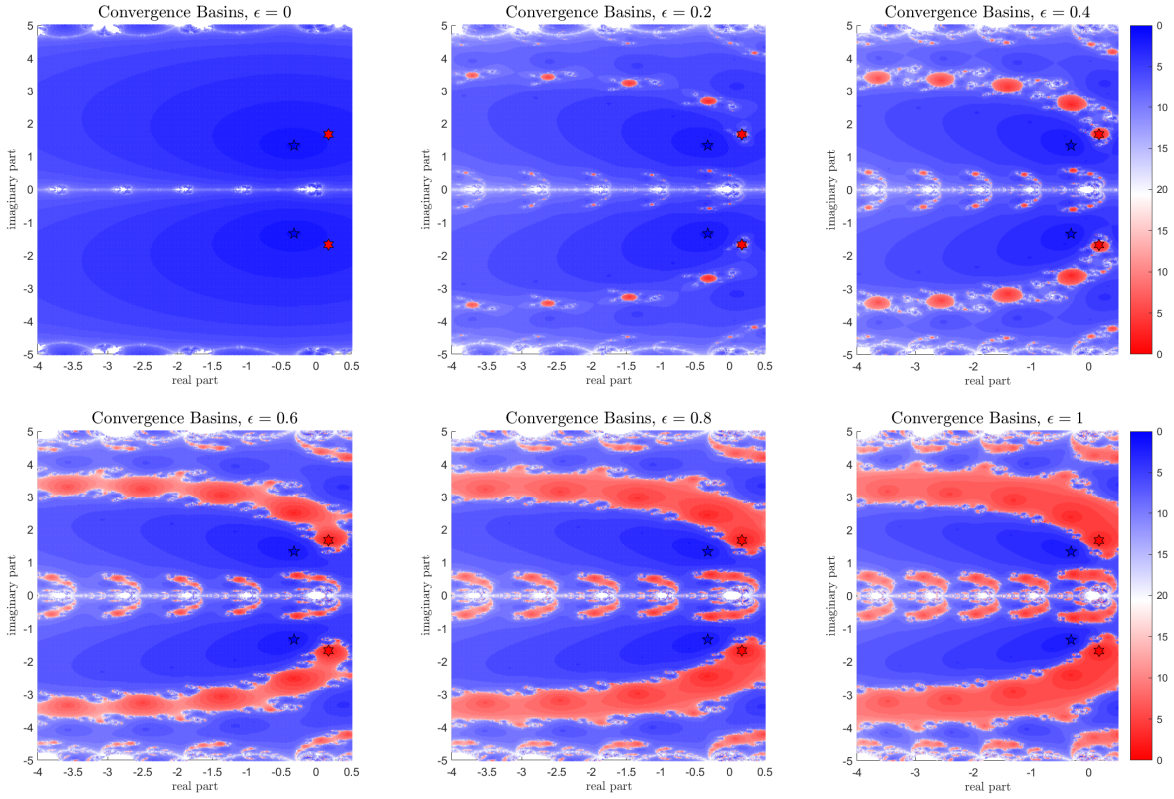

Figure 5.3: Basins of convergence for iteration (5.2) for $f(z)$ defined in (5.8).The shading of each point is determined by the number of iterations needed to converge, taken by when the relative change in z_k was less than 10^{-5}. Blue coloration indicates convergence to a root of $d_j(z)$, and red coloration indicate convergence to a root of $d_q(z)$. White denotes that the iteration did not converge in 20 steps. We show basins for $\epsilon = 0, 0.2, \ldots, 1$.

We can also note that the behavior of the convergence of the iteration defined by (5.2) differs depending on how the nonlinear eigenvalue problem is defined. We can pose the simple delay problem in the form $\mathbf{T}(z)\mathbf{v} = \mathbf{0}$ in two natural ways, either as

$$\mathbf{T}_1(z) = z\mathbf{I} - e^{-\tau z}\mathbf{A}$$

or as

$$\mathbf{T}_2(z) = ze^{\tau z}\mathbf{I} - \mathbf{A}.$$

We have here that $\mathbf{T}_2(z) = e^{\tau z}\mathbf{T}_1(z)$. We see that $\sigma(\mathbf{T}_1) = \sigma(\mathbf{T}_2)$ given that $e^{\tau z} \neq 0$ for any $z \in \mathbb{C}$. However, these two problems exhibit different iteration behaviors. We see that in these two cases the scalar function f defined in (5.8) will change, where

$$d_k(z) = z - e^{-\tau z}\lambda_k,$$

when working with $\mathbf{T}_1$, and

$$\widetilde{d}_k(z) = ze^{\tau z} - \lambda_k$$

when working with $\mathbf{T}_2$, where $\lambda_k \in \sigma(\mathbf{A})$. Using the same computational experiment, we see different basins of attraction, with much larger regions where the method failed to converge.

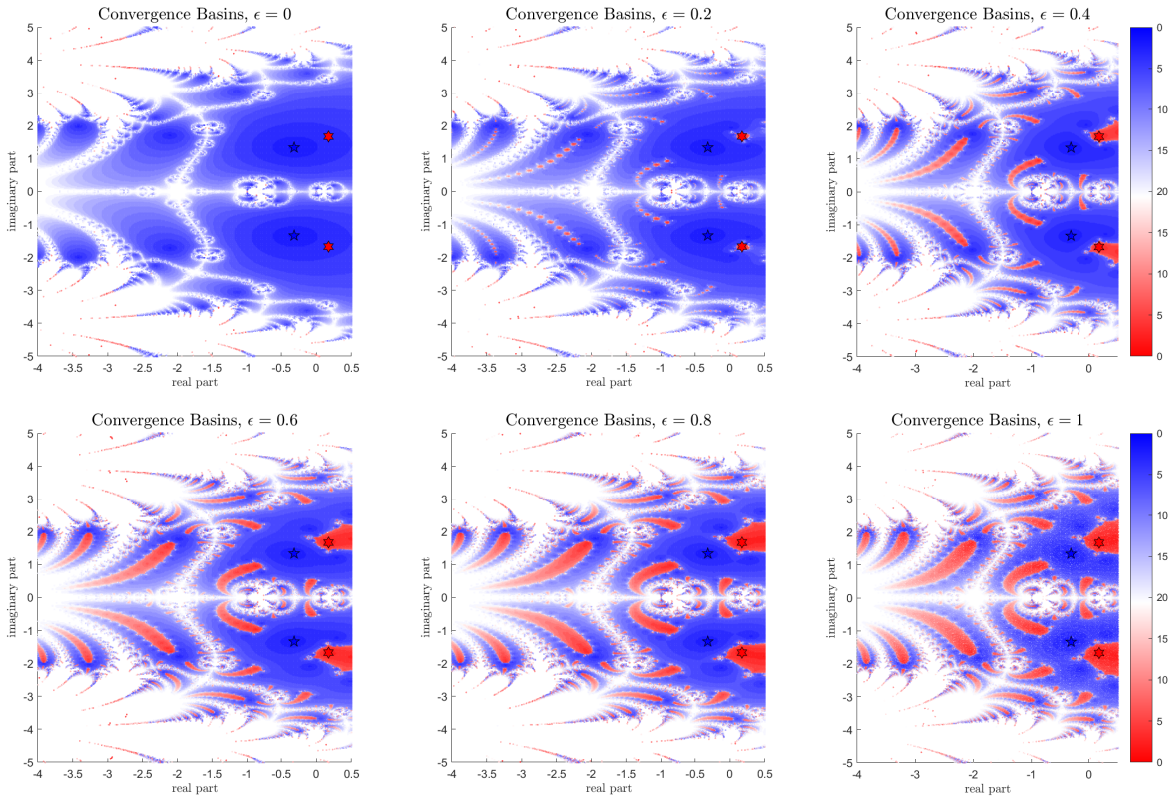

Figure 5.4: Basins of convergence for iteration (5.2) for $f(z)$ defined in (5.8). The shading of each point is determined by the number of iterations needed to converge, taken by when the relative change in z_k was less than 10^{-5}. Blue coloration indicates convergence to a root of $\widetilde{d}_j(z)$, and red coloration indicate convergence to a root of $\widetilde{d}_q(z)$. White denotes that the iteration did not converge in 20 steps. We show basins for $\epsilon = 0, 0.2, \ldots, 1$.

Now we discuss how singular vectors can be used as interpolation directions, given that it is not reasonable to expect to know $\mathbf{W}$ or $\mathbf{V}$.

5.3.4 Updating Interpolation Directions with Singular Vectors

Recall that (5.2) was derived based on one updated interpolation point and constant interpolation directions. Now let us assume we update the pair of interpolation directions as well. In this case we could write the recurrence formula as

$$z_{i+1} = z_i + \frac{f_i(z_i)}{f_i'(z_i)}, \tag{5.9}$$

where $f_i(z) = \boldsymbol{\ell}^{(i)^*}\mathbf{T}(z)^{-1}\mathbf{r}^{(i)}$, noting that the directions now depend on the iteration step. Assume we wish to interpolate at the point μ, and that $\mathbf{T}(\mu)$ has the singular value decomposition,

$$\mathbf{T}(\mu) = \mathbf{Y}\boldsymbol{\Sigma}\mathbf{X}^* = \sum_{j=1}^{n} s_j \mathbf{y}_j \mathbf{x}_j^*.$$

If $\mu \notin \sigma(\mathbf{T})$, then $\mathbf{T}(\mu)$ is nonsingular, and so $s_j \neq 0$, for all j. Then note that

$$\mathbf{T}(\mu)^{-1} = \sum_{j=1}^{n} \frac{1}{s_j} \mathbf{x}_j \mathbf{y}_j^*$$

and so

$$f_i(\mu) = \boldsymbol{\ell}^{(i)^*}\mathbf{T}(\mu)^{-1}\mathbf{r}^{(i)} = \sum_{j=1}^{n} \frac{(\boldsymbol{\ell}^{(i)^*}\mathbf{x}_j)(\mathbf{y}_j^*\mathbf{r}^{(i)})}{s_j}.$$

If we take $\boldsymbol{\ell}^{(i)} = \mathbf{x}_q$ and $\mathbf{r}^{(i)} = \mathbf{u}_q$, where $1 \leq q \leq n$, then

$$f_i(\mu) = \frac{1}{s_q}.$$

Using an identity for the derivative of the inverse of a matrix-valued function, we have

$$(\mathbf{T}(\mu)^{-1})' = -\mathbf{T}(\mu)^{-1}\mathbf{T}'(\mu)\mathbf{T}(\mu)^{-1},$$

and therefore,

$$\begin{aligned} f_i'(\mu) &= \boldsymbol{\ell}^{(i)^*}(\mathbf{T}(\mu)^{-1})'\mathbf{r}^{(i)} \\ &= -\boldsymbol{\ell}^{(i)^*}\mathbf{T}(\mu)^{-1}\mathbf{T}'(\mu)\mathbf{T}(\mu)^{-1}\mathbf{r}^{(i)} \\ &= -\left(\sum_{j=1}^{n}\frac{(\boldsymbol{\ell}^{(i)^*}\mathbf{x}_j)\mathbf{y}_j^*}{s_j}\right)\mathbf{T}'(\mu)\left(\sum_{k=1}^{n}\frac{\mathbf{x}_k(\mathbf{u}_k^*\mathbf{r}^{(i)})}{s_k}\right) \\ &= -\frac{\mathbf{u}_q^*\mathbf{T}'(\mu)\mathbf{x}_q}{s_q^2}. \end{aligned}$$

Plugging this result into (5.9), we obtain,

$$z_{i+1} = z_i - \frac{s_q}{\mathbf{u}_q^*\mathbf{T}'(z_i)\mathbf{x}_q} = z_i - \frac{\mathbf{u}_q^*\mathbf{T}(z_i)\mathbf{x}_q}{\mathbf{u}_q^*\mathbf{T}'(z_i)\mathbf{x}_q}. \tag{5.10}$$

If we take $q = n$, so that $s_q = s_n$ is the smallest singular value, then as $s_q \to 0$, we have that $z_i \to \lambda \in \sigma(\mathbf{T}(z))$, assuming $\mathbf{u}_q^*\mathbf{T}'(z_i)\mathbf{v}_q$ does not go to zero as well. This method is summarized in Algorithm 8.

Algorithm 8 IRI-1 using singular vectors as interpolation directions

input: initial interpolation point $z_0 \in \mathbb{C}$.

for $k = 0, 1, \ldots$ until converged **do**

1. Compute smallest singular value (s) and corresponding singular vectors ($\mathbf{u}, \mathbf{x}$) of
2. $\mathbf{T}(z_k)$.
3. Set $f_k = s$ and $f_k' = \mathbf{u}^*\mathbf{T}'(z_k)\mathbf{x}$
4. Set $z_{k+1} = z_k + \dfrac{f_k}{f_k'}$.

end for

Figure 5.5 shows results for Algorithm 8 with three different initial starting points. Convergence was determined when the relative change in the interpolation point was less than 10^{-5}.

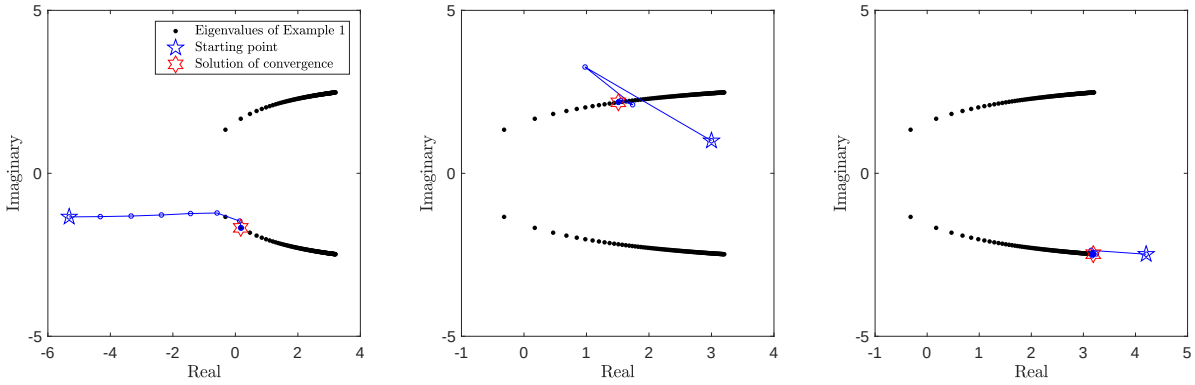

Figure 5.5: Results of Algorithm 8: (left) iteration took 10 step to converge; (center) iteration took 7 step to converge; (right) iteration took 10 step to converge

Notably in column 1 of Figure 5.5, we see Algorithm 8 does not converge to an eigenvalue close to an intermediate interpolation point. We now expand Algorithm 8 when using multiple interpolation points. Due to the interactions between the interpolation directions when forming the Loewner matrices, we are unable to form a simple expression similar to (5.10). This method is summarized in Algorithm 9.

Algorithm 9 IRI-multiple point using singular vectors as interpolation directions

input: an initial set of interpolation points $\{z_{0,j}\}_{j=1}^{r} \subset \mathbb{C}$.

for $k = 0, 1, \ldots$ until converged **do**

for $j = 1, 2, \ldots r$ **do**

(i) Compute smallest singular value (s_j) and corresponding singular vectors ($\mathbf{u}_j, \mathbf{x}_j$) of $\mathbf{T}(z_{k,j})$.

(ii) Set interpolation directions to singular vectors:

$$\boldsymbol{\ell}_j \leftarrow \mathbf{y}_j, \quad \mathbf{r}_j \leftarrow \mathbf{x}_j.$$

end for

1. Compute Loewner and shifted Loewner matrices as detailed in Theorem 3.1.1.

2. Compute eigenvalues of Loewner pencil

$$\{\mu_1, \ldots, \mu_r\} = \sigma(\mathbb{L}_s, \mathbb{L}).$$

3. Set $z_{k+1,j} = \mu_j$ for $j = 1, 2, \ldots r$.

end for

Results for Algorithm 9 applied to the simple delay problem of Example 1 are shown in Figure 5.6. Convergence was measured by the relative change in the interpolation points,

$$\frac{\|\mathbf{z}_{k+1} - \mathbf{z}_k\|_2}{\|\mathbf{z}_{k+1}\|_2}$$

where $\mathbf{z}_{k+1}$ is the vector of interpolation points $z_{k,j}$ for $j = 1, \ldots, r$, with a relative tolerance

of 10^{-5}. Here several interpolation points converged to the same solution, which is why two approximations are shown.

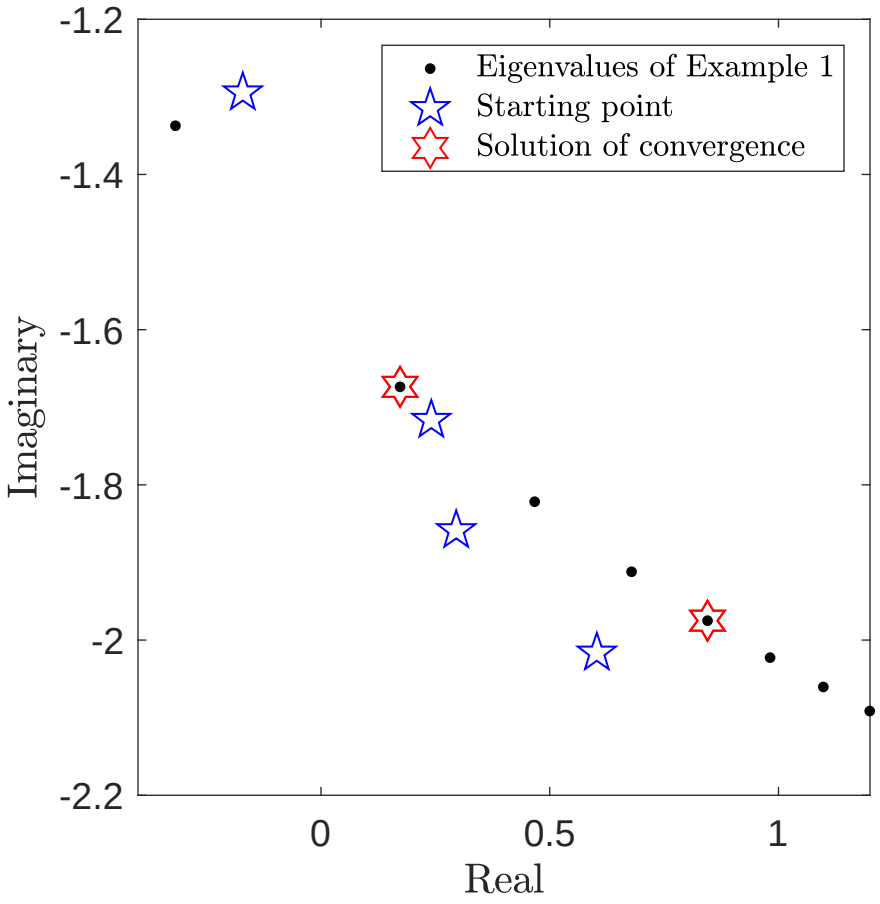

Figure 5.6: Results for Algorithm 9. The 4 initial interpolation points were taken as the interpolation points used in Chapter 2 while discussing polynomial linearization.

5.3.5 Updating Interpolation Directions using Approximate Eigenvectors

We can also use the approximate eigenvectors generated by the rational interpolant to update the interpolation directions. Recall that if $(\lambda_j, \mathbf{x}_j, \mathbf{y}_j)$ is an eigen-triple of the Loewner pencil $(\mathbb{L}_s, \mathbb{L})$, then $(\lambda_j, \mathbf{C}_r\mathbf{x}_j, \mathbf{B}_r^*\mathbf{y}_j)$ is an approximate eigen-triple of $\mathbf{T}$. This methodology in fact imitates the iterative rational Krylov algorithm using transfer function evaluations (TF-IRKA) found in [12], with the notable difference being that the interpolation points are updated to be the eigenvalues of the Loewner pencil, rather than their reflection across the imaginary axis. In the case where $r = 1$, the eigenvectors of the Loewner pencil reduce to scalars, and therefore the approximate eigenvectors of $\mathbf{T}$ are in fact the vectors of left and right data. Algorithm 10 summarizes this method.

Algorithm 10 IRI-1 Using approximate eigenvectors as interpolation directions

input: initial interpolation point $z_0 \in \mathbb{C}$, and set left/right interpolation directions $\boldsymbol{\ell}, \mathbf{r} \in \mathbb{C}^n$.

for $k = 0, 1, \ldots$ until converged **do**

1. Compute $f_k = \boldsymbol{\ell}^*\mathbf{T}(z_k)^{-1}\mathbf{r}$ and $f_k' = \boldsymbol{\ell}^*\mathbf{T}(z_k)^{-1}\mathbf{T}'(z_k)\mathbf{T}(z_k)^{-1}\mathbf{r}$
2. Set $z_{k+1} = z_k + \dfrac{f_k}{f_k'}$.
3. Set

$$\boldsymbol{\ell} \leftarrow \mathbf{b} = \mathbf{T}(z_k)^{-*}\boldsymbol{\ell}, \quad \mathbf{r} \leftarrow \mathbf{c} = \mathbf{T}(z_k)^{-1}\mathbf{r}$$

end for

Figure 5.7 show results for Algorithm 10 with three different initial starting points. Converge was determined when the relative change in the interpolation point was less then 10^{-5}. Comparing to the results of Algorithm 8. we see in column 1, the iteration does converge to the left most eigenvalue.

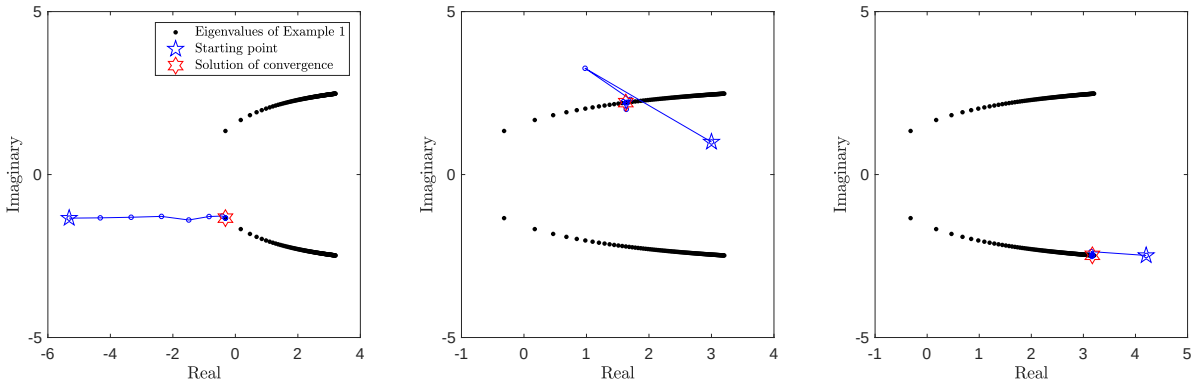

Figure 5.7: Results of Algorithm 10: (left) iteration took 10 step to converge; (center) iteration took 8 step to converge; (right) iteration took 10 step to converge

We now expand Algorithm 10 when using multiple interpolation points, summarized in Algorithm 11.

Algorithm 11 IRI-multiple points using approximate eigenvectors as interpolation directions

input: interpolation points $\{z_{0,j}\}_{j=1}^{r} \subset \mathbb{C}$, and left/right interpolation directions $\{\boldsymbol{\ell}_j\}_{j=1}^{r}, \{\mathbf{r}_j\}_{j=1}^{r}, \subset \mathbb{C}^n$:

for $k = 0, 1, \ldots$ until converged **do**

1. Compute Loewner matrices $\mathbb{L}$ and $\mathbb{L}_s$, and matrices of left and right data $\mathbf{B}_r$ and $\mathbf{C}_r$ as detailed in Theorem 3.1.1 using $\mathbf{T}(z)^{-1}$.
2. Obtain eigen-triples of the Loewner pencil $(\mathbb{L}_s, \mathbb{L})$ as $(\lambda_j, \mathbf{x}_j, \mathbf{y}_j)$ for $j = 1, \ldots, m$
3. Set

$$\boldsymbol{\ell}_j \leftarrow \mathbf{B}_r^* \mathbf{y}_j, \quad \mathbf{r}_j \leftarrow \mathbf{C}_r \mathbf{x}_j, \quad z_{k+1,j} = \lambda_j.$$

end for

We now provide results for Algorithm 11 applied to the simple decay problem of Example 1. Convergence was measured by the relative change in the interpolation points,

$$\frac{\|\mathbf{z}_{k+1} - \mathbf{z}_k\|}{\|\mathbf{z}_{k+1}\|}$$

where $\mathbf{z}_{k+1}$ is the vector of interpolation points $z_{k,j}$ for $j = 1, \ldots, r$, with a relative tolerance of 10^{-3}. Here several interpolation points converged to the same solution, which is why two approximations are shown.

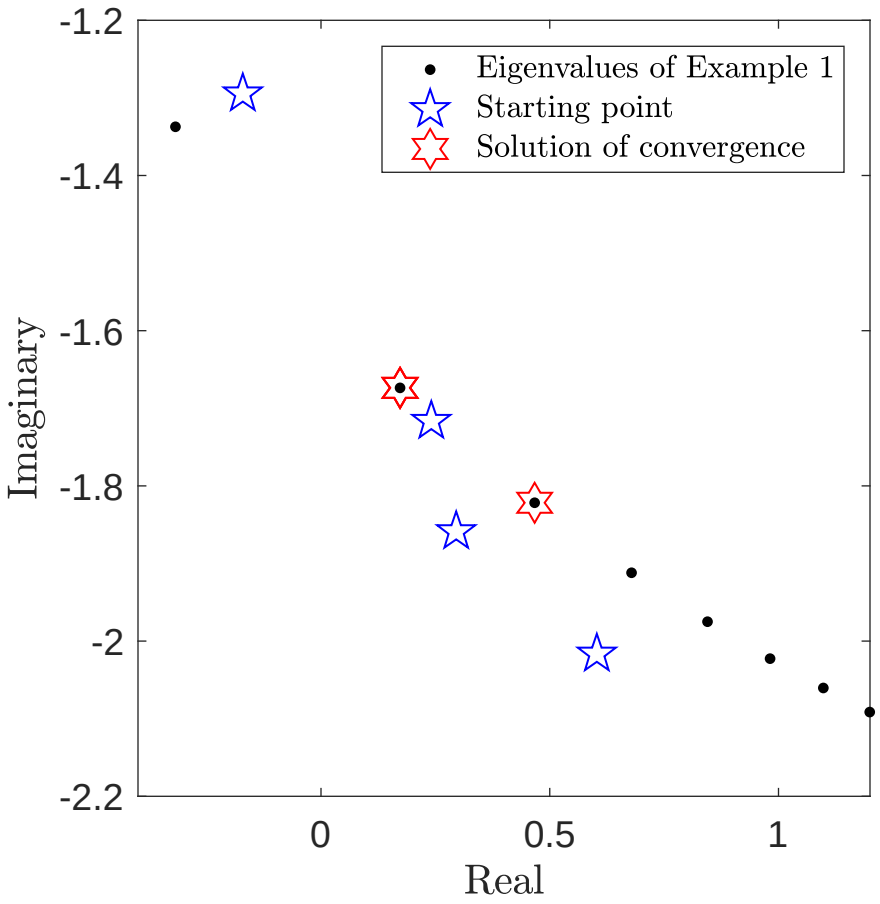

Figure 5.8: Results for Algorithm 11 for 12 iterations: the 4 initial interpolation points were taken as the interpolation points used in Chapter 2 while discussing polynomial linearization.

Chapter 6

Conclusions

This thesis has proposed several contributions to current methods for nonlinear eigenvalue problems. The first contribution of this work connected recent contour integration methods to the theory and practice of system identification. This observation led us to expand the methodology of [10, 15] by using general rational interpolation for system identification, producing a Loewner matrix contour integration technique. The second development of this work studied the application of rational interpolation directly to the function $\mathbf{T}(z)^{-1}$, and we found that the poles of this interpolant approximate the eigenvalues of $\mathbf{T}$. We then expanded this idea to several iterative methods, where at each step the approximate eigenvalues are taken as new interpolation points. We showed that the case where one interpolation point is used is theoretically equivalent to Newton's method for a particular scalar function.

Future Work

Several future research directions are planned for the methods proposed in this thesis. It will be beneficial to study the effects of different interpolation point locations on the accuracy and numerical stability of the Loewner contour integration method. We showed preliminary results suggesting that the placement of the interpolation points alter an underlying filter function, and that we may be able to lessen numerical contributions from eigenvalues exterior to a given search region. Further development of this process will aid in situations where eigenvalues are clustered close together. We will also investigate the success of the Loewner contour integration method when taking the interpolation points inside the search region.

Bibliography

[1] A. Amiraslani, R. M. Corless, and P. Lancaster, *Linearization of matrix polynomials expressed in polynomial bases*, IMA Journal of Numerical Analysis, 29 (2008), pp. 141–157.

[2] A. L. Andrew, K. E. Chu, and P. Lancaster, *Derivatives of eigenvalues and eigenvectors of matrix functions*, SIAM Journal on Matrix Analysis and Applications, 14 (1993), pp. 903–926.

[3] ——, *On the numerical solution of nonlinear eigenvalue problems*, Computing, 55 (1995), pp. 91–111.

[4] P. Anselone and L. Rall, *The solution of characteristic value-vector problems by Newton's method*, Numerische Mathematik, 11 (1968), pp. 38–45.

[5] A. C. Antoulas, *Approximation of Large-Scale Dynamical Systems*, Philadelphia, PA: Siam, 2005.

[6] A. C. Antoulas and B. Anderson, *On the scalar rational interpolation problem*, IMA Journal of Mathematical Control and Information, 3 (1986), pp. 61–88.

[7] A. C. Antoulas, C. A. Beattie, and S. Gugercin, *Interpolatory model reduction of large-scale dynamical systems*, in Efficient Modeling and Control of Large-Scale Systems, New York, NY: Springer, 2010, pp. 3–58.

[8] A. C. Antoulas, I. V. Gosea, and A. C. Ionita, *Model reduction of bilinear systems in the Loewner framework*, SIAM Journal on Scientific Computing, 38 (2016), pp. B889–B916.

[9] A. C. Antoulas, S. Lefteriu, and A. C. Ionita, *A tutorial introduction to the Loewner framework for model reduction*, in Model Reduction and Approximation: Theory and Algorithms, edited by: P. Benner, M. Ohlberger, A. Cohen & K. Willcox, Series: Computational Science Engineering, (2014), pp. 335–376.

[10] J. Asakura, T. Sakurai, H. Tadano, T. Ikegami, and K. Kimura, *A numerical method for nonlinear eigenvalue problems using contour integrals*, JSIAM Letters, 1 (2009), pp. 52–55.

[11] C. Beattie and S. Gugercin, *Interpolatory projection methods for structure-preserving model reduction*, Systems & Control Letters, 58 (2009), pp. 225–232.

[12] ——, *Realization-independent H2-approximation*, in 2012 IEEE 51st Annual Conference on Decision and Control (CDC), IEEE, 2012, pp. 4953–4958.

[13] T. Betcke, N. J. Higham, V. Mehrmann, C. Schroder, and F. Tisseur, *NLEVP: A collection of nonlinear eigenvalue problems*, ACM Transactions on Mathematical Software (TOMS), 39 (2013), p. 7.

[14] T. Betcke and H. Voss, *A Jacobi–Davidson-type projection method for nonlinear eigenvalue problems*, Future Generation Computer Systems, 20 (2004), pp. 363–372.

[15] W.-J. Beyn, *An integral method for solving nonlinear eigenvalue problems*, Linear Algebra and its Applications.

[16] R. M. Corless, G. H. Gonnet, D. E. Hare, D. J. Jeffrey, and D. E. Knuth, *On the Lambert-W function*, Advances in Computational Mathematics, 5 (1996), pp. 329–359.

[17] C. K. Garrett, Z. Bai, and R.-C. Li, *A nonlinear QR algorithm for banded nonlinear eigenvalue problems*, ACM Transactions on Mathematical Software (TOMS), 43 (2016), p. 4.

[18] I. Gohberg, P. Lancaster, and L. Rodman, *Matrix Polynomials*, Springer, 2005.

[19] I. V. Gosea and A. C. Antoulas, *Data-driven model order reduction of quadratic-bilinear systems*, Numerical Linear Algebra with Applications, (2018), p. e2200.

[20] S. Gugercin, A. C. Antoulas, and C. Beattie, *H2 model reduction for large-scale linear dynamical systems*, SIAM Journal on Matrix Analysis and Applications, 30 (2008), pp. 609–638.

[21] S. Guttel and F. Tisseur, *The nonlinear eigenvalue problem*, Acta Numerica, 26 (2017), p. 194.

[22] S. Guttel, R. Van Beeumen, K. Meerbergen, and W. Michiels, *NLEIGS: A class of fully rational Krylov methods for nonlinear eigenvalue problems*, SIAM Journal on Scientific Computing, 36 (2014), pp. A2842–A2864.

[23] M. V. Keldysh, *On the characteristic values and characteristic functions of certain classes of non-self-adjoint equations*, in Doklady Akad. Nauk SSSR (NS), vol. 77, 1951, pp. 11–14.

[24] ——, *On the completeness of the eigenfunctions of some classes of non-selfadjoint linear operators*, Russian Mathematical Surveys, 26 (1971), pp. 15–44.

[25] V. Khazanov and V. Kublanovskaya, *Spectral problems for matrix pencils. Methods and algorithms. I.*, Russian Journal of Numerical Analysis and Mathematical Modelling, 3 (1988), pp. 337–372.

[26] ——, *Spectral problems for matrix pencils. Methods and algorithms. II.*, Russian Journal of Numerical Analysis and Mathematical Modelling, 3 (1988), pp. 467–486.

[27] P. Kunkel and V. Mehrmann, *Differential-Algebraic Equatios: Analysis and Numerical Solution*, vol. 2, Zurich, Switzerland: European Mathematical Society, 2006.

[28] P. Lancaster, *A generalised Rayleigh quotient iteration for lambda-matrices*, Archive for Rational Mechanics and Analysis, 8 (1961), p. 309.

[29] P. Lancaster, *Lambda-matrices and Vibrating Systems*, Oxford, England: Pergamon, 1966.

[30] P. Lietaert, J. Pérez, B. Vandereycken, and K. Meerbergen, *Automatic rational approximation and linearization of nonlinear eigenvalue problems*, arXiv preprint arXiv:1801.08622, (2018).

[31] A. Mayo and A. C. Antoulas, *A framework for the solution of the generalized realization problem*, Linear Algebra and its Applications, 425 (2007), pp. 634–662.

[32] V. Mehrmann and D. Watkins, *Polynomial eigenvalue problems with Hamiltonian structure*, Electronic Transactions on Numerical Analysis, 13 (2002), pp. 106–118.

[33] W. Michiels and S.-I. Niculescu, *Stability and Stabilization of Time-Delay Systems: An Eigenvalue-Based Approach*, Philadelphia, PA:SIAM, 2007.

[34] Y. Nakatsukasa, O. Sète, and L. N. Trefethen, *The AAA algorithm for rational approximation*, SIAM Journal on Scientific Computing, 40 (2018), pp. A1494–A1522.

[35] A. Neumaier, *Residual inverse iteration for the nonlinear eigenvalue problem*, SIAM Journal on Numerical Analysis, 22 (1985), pp. 914–923.

[36] J. Rommes and G. L. Sleijpen, *Convergence of the dominant pole algorithm and Rayleigh quotient iteration*, SIAM Journal on Matrix Analysis and Applications, 30 (2008), pp. 346–363.

[37] A. Ruhe, *Algorithms for the nonlinear eigenvalue problem*, SIAM Journal on Numerical Analysis, 10 (1973), pp. 674–689.

[38] K. Schreiber, *Nonlinear eigenvalue problems: Newton-type methods and nonlinear Rayleigh functionals*, Doctoral Thesis, Technische Universitat Berlin, (2008).

[39] H. Schwetlick and K. Schreiber, *Nonlinear Rayleigh functionals*, Linear Algebra and its Applications, 436 (2012), pp. 3991–4016.

[40] L. Silverman, *Realization of linear dynamical systems*, IEEE Transactions on Automatic Control, 16 (1971), pp. 554–567.

[41] K. Sinani, S. Gugercin, and C. Beattie, *A structure-preserving model reduction algorithm for dynamical systems with nonlinear frequency dependence*, IFAC-PapersOnLine, 49 (2016), pp. 56–61.

[42] F. Tisseur and K. Meerbergen, *The quadratic eigenvalue problem*, SIAM Review, 43 (2001), pp. 235–286.

[43] L. N. Trefethen and J. Weideman, *The exponentially convergent trapezoidal rule*, SIAM Review, 56 (2014), pp. 385–458.

[44] M. Van Barel, *Designing rational filter functions for solving eigenvalue problems by contour integration*, Linear Algebra and its Applications, 502 (2016), pp. 346–365.

[45] M. Van Barel and P. Kravanja, *Nonlinear eigenvalue problems and contour integrals*, Journal of Computational and Applied Mathematics, 292 (2016), pp. 526–540.

[46] R. Van Beeumen, K. Meerbergen, and W. Michiels, *Compact rational Krylov methods for nonlinear eigenvalue problems*, SIAM Journal on Matrix Analysis and Applications, 36 (2015), pp. 820–838.

[47] R. Van Beeumen, W. Michiels, and K. Meerbergen, *Linearization of Lagrange and Hermite interpolating matrix polynomials*, IMA Journal of Numerical Analysis, 35 (2015), pp. 909–930.

[48] H. Voss, *An Arnoldi method for nonlinear eigenvalue problems*, BIT numerical mathematics, 44 (2004), pp. 387–401.

[49] H. Voss, *A Jacobi–Davidson method for nonlinear and nonsymmetric eigenproblems*, Computers & Structures, 85 (2007), pp. 1284–1292.

[50] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, vol. 87, Oxford, England: Clarendon Press, 1965.