

MULTI INPUT MINIMAX ADAPTIVE ANTOULAS–ANDERSON ALGORITHM  
FOR RATIONAL APPROXIMATION WITH STABLE POLES

by

William Richard Johns

A dissertation submitted in partial fulfillment  
of the requirements for the degree

of

Doctor of Philosophy

in

Mathematics

MONTANA STATE UNIVERSITY  
Bozeman, Montana

April 2021

©COPYRIGHT

by

William Richard Johns

2021

All Rights Reserved

## ACKNOWLEDGEMENTS

I sincerely thank and acknowledge the following people without whom this thesis would never have been conceived or completed.

Dr. Bjorn Gustavson, SINTEF, for answering many emails about Vector Fitting.

Dr. Stefan Güttel, University of Manchester, for multiple interesting and informative conversations about RKFIT.

Dr. Françoise Tisseur, University of Manchester, for answering many emails about weighted set-valued AAA and non-linear eigenvalue problems.

Dr. Matthew Reynolds, NREL, for being my mentor during the internship where this thesis research began and during the NSF INTERN program (DMS #1748883) which supported this work.

Dr. Lucas Monzón, University of Colorado Boulder, for tirelessly working on this project with me despite his ever busy life and, my being a student at a different university altogether.

Dr. Lukas Geyer and Dr. Dominique Zosso, Montana State University, for agreeing to be readers on my thesis and for all their help with the complex analysis and optimization that went into this work.

Dr. Lisa Davis, Montana State University, for encouragement, advice and agreeing to be the chair of my committee. For working through every aspect of this project with me despite my regular abuse of notation and terrible handwriting.

This work was supported by the MT PEAKS program (DMS #1748883)

## TABLE OF CONTENTS

1. INTRODUCTION .....	1
1.0.1 Benchmark Example: EMT .....	3
1.0.2 Summary .....	3
2. RATIONAL APPROXIMATIONS AND THEIR REPRESENTATIONS .....	6
2.1 Single Input Single Output Rational Approximations .....	6
2.2 Partial Fraction and Barycentric Representations .....	8
2.3 Multi-Input Multi-Output Rational Approximations .....	10
2.4 Vector Norms .....	11
2.5 Applications of MIMO Rational Approximations .....	12
2.5.1 Linear Time-Invariant Dynamical Systems .....	12
2.5.2 Nonlinear Eigenvalue Problems .....	14
3. CURRENT ALGORITHMS .....	16
3.1 Current Algorithms for MIMO Rational Approximation .....	16
3.1.1 The Vector Fitting Algorithm .....	17
3.1.2 RKFIT .....	18
3.2 The AAA Algorithm and Barycentric Lawson Optimization .....	20
3.2.1 AAA Algorithm .....	21
3.2.2 Lawson Optimization of Barycentric Interpolants .....	23
4. DEVELOPMENT OF MULTI-INPUT MULTI-OUTPUT AAA .....	27
4.1 Multi-Input Multi-Output Extension of AAA .....	27
4.1.1 Example Failure of Linearization .....	29
4.1.2 Example: miAAA on Six Functions .....	32
4.1.3 FastAAA .....	32
4.1.4 Hermitian Symmetry in LTI Applications .....	34
4.1.5 Set-Valued AAA .....	36
5. METHODOLOGY FOR STABILITY ENFORCEMENT .....	37
5.1 Stability and Stable miAAA .....	37
5.1.1 Stable miAAA Example .....	40
5.1.2 Experimental Observations on the Reflection of the Poles .....	42
5.1.3 Enforcing Other Pole Constraints .....	43

## TABLE OF CONTENTS – CONTINUED

6. LAWSON OPTIMIZATIONS .....	46
6.1 MIMO Lawson Optimization .....	46
6.1.1 MIMO Lawson Example .....	48
6.2 Stable Lawson Optimization .....	49
6.2.1 Stable MIMO Lawson Example .....	52
7. CONVERSION TO PARTIAL FRACTIONS.....	54
7.1 Computation of the Poles.....	54
7.1.1 Case I: When $\deg(P)=N - 1$ .....	55
7.1.2 Case II: When $\deg(P)< N - 1$ .....	56
7.1.3 Deflating the Spurious Eigenvalues .....	57
7.2 Computation of the Residues.....	58
7.3 Computation of the Polynomial Coefficients .....	59
7.3.1 Conversion to Partial Fraction Example .....	60
7.3.2 Effects of Singular Perturbation.....	60
8. MODEL ORDER REDUCTION TEST PROBLEMS.....	63
8.1 Example: Vector Fitting Users Guide.....	64
8.2 Example:Matrix Fitting Toolbox .....	66
8.3 Example: International Space Station .....	68
9. CONCLUSIONS AND FUTURE WORK .....	71
REFERENCES CITED.....	73

## LIST OF TABLES

Table	Page
7.1 Poles of $\vartheta$ as computed with <i>prz</i> .....	62
7.2 Poles of $\vartheta$ as computed with <i>przd</i> .....	62
8.1 Errors and running times for different algorithms on the Vector Fitting Users Guide example with 67 poles .....	65
8.2 Errors and running times for different algorithms on MFT example with 55 poles .....	67
8.3 Errors and running times for different algorithms on the international Space Station example with 49 poles .....	69

## LIST OF FIGURES

Figure	Page
1.1 Logarithmic absolute value of the 6 entries from the admittance matrix.....	4
1.2 Schematic of MIMO power network from [19], with cable lengths given in km.....	4
4.1 Errors for miAAA approximation of the six entries with input tolerance of $10^{-6}$ .....	33
5.1 Logarithmic errors for smiAAA approximation of the six entries with input tolerance of $10^{-6}$ .....	41
5.2 Logarithmic errors for smiAAA approximation at each iteration with different <i>ref</i> values plotted using different colors and tolerance $10^{-8}$ .....	42
5.3 Location of poles chosen obtained by set-valued AAA (blue) vs. smiAAA (red) for a nonlinear eigenvalue problem.....	44
5.4 Location of poles (enlarged) obtained by set-valued AAA (blue) and smiAAA (red) for a nonlinear eigenvalue problem.....	45
6.1 Max error $\max_k  f_k(z) - B_k(z) $ , $k = 1, 2, \dots, 6$ , for miAAA approximation of the six entries from the example in Section 4.1.2 with input tolerance of $10^{-6}$ before and after ten iterations of the MIMO Lawson optimization.....	50
6.2 Max error $\max_k  f_k(z) - B_k(z) $ , $k = 1, 2, \dots, 6$ for smiAAA approximation of the six entries with input tolerance of $10^{-6}$ before and after ten iterations of the stable MIMO Lawson optimization.....	53
7.1 Logarithmic errors for smiAAA approximation of the six entries with input tolerance of $10^{-6}$ in barycentric form vs partial fraction form .....	61
8.1 Errors for different algorithms on Vector Fitting Users Guide example.....	65

## LIST OF FIGURES – CONTINUED

Figure	Page
8.2 Number of poles chosen by symmetric smiAAA and FastAAA for different accuracies on VF users guide example .....	66
8.3 Errors for different algorithms on MFT example .....	67
8.4 Number of poles chosen by symmetric smiAAA and FastAAA for different accuracies on the MTF example .....	68
8.5 Errors for different algorithms on the International Space Station example.....	70
8.6 Errors for different algorithms on the ISS example .....	70

## LIST OF ALGORITHMS

Algorithm	Page
3.1 AAA .....	24
3.2 Barycentric Lawson Optimization.....	26
4.3 miAAA .....	30
5.4 smiAAA .....	40
6.5 MIMO Barycentric Lawson Optimization.....	49
6.6 Stable MIMO Barycentric Lawson Optimization .....	52

## NOMENCLATURE

$r$	Rational function
$P$	Numerator polynomial of rational function
$Q$	Denominator polynomial of rational function
$B$	The barycentric interpolant with support points $\mathbf{z}$ and weights $\mathbf{w}$
$J$	Numerator of barycentric representation
$D$	Denominator of barycentric representation
$\mathbf{Z}$	The set of Common Sample points
$\mathbf{z}$	The set of Common Support points
$\mathbf{w}$	The interpolatory barycentric weights
$N$	The Number of chosen Support Points
$M$	The number of Sample points
$Z^{(j)}$	The set of common sample points not chosen as support points at iteration N
$pf$	The partial fraction representation
$\mathbf{lw}$	The Lawson weights during Lawson- type optimization
$\mathbf{w}^\wedge$	The non-interpolatory barycentric numerator weights
$\mathbf{w}^\vee$	The non-interpolatory barycentric denominator weights
$B^\wedge$	The non-interpolatory barycentric interpolant
$C$	The Cauchy matrix for computing the interpolatory barycentric weights
$C^l$	The Cauchy matrix for computing the non-interpolatory barycentric weights

## ABSTRACT

This thesis details the development of the “symmetric stable multi-input multi-output Adaptive Antoulas Anderson” algorithm, we call this algorithm symmetric smiAAA. The symmetric smiAAA algorithm builds rational approximations, for multiple inputs. The approximations share a common set of parameters called the poles. The primary goal of this algorithm is to address shortcomings in multi-input multi-output rational approximation algorithms currently used in electro-magnetic transients programs.

All state of the art algorithms currently follow a similar methodology: The user selects the number of poles to use and supplies an initial guess for their values. The algorithms optimize the shared poles and return the best approximation they found. The user is not guaranteed a specific accuracy in the approximations. If the results returned are not sufficiently accurate, the algorithm must be run again with additional poles. Symmetric smiAAA is designed with the goal of achieving user-defined accuracy, with no information about the number of poles. The user selects the desired accuracy of the approximations and the algorithm does the rest. Symmetric smiAAA returns approximations with the desired accuracy by finding the number of shared poles needed for the desired accuracy, and their values.

This work introduces the following three features to the “Adaptive Antoulas Anderson” algorithm. First, we extend the ideas from the single-input Adaptive Antoulas Anderson algorithm, to multi-input multi-output problems. Second, we introduce enforcement of constraints on the values of the poles. Lastly, we extend a single input post-processing optimization based upon the Lawson method, to multi-input multi-output problems.

The symmetric smiAAA algorithm combines these three features with the symmetry enforcement introduced in the FastAAA algorithm. In order to test it against the current industry standards, we compare the symmetric smiAAA algorithm with Vector Fitting and the recently published RKFIT algorithms. These comparisons demonstrate that symmetric smiAAA produces approximations with similar accuracy and running time, while allowing the user to select only the desired accuracy. Symmetric smiAAA is a robust and powerful algorithm which provides the user full control over the final accuracy of the approximations.

## CHAPTER ONE

## INTRODUCTION

This research began during a summer internship at the National Renewable Energy Lab (NREL). The work in this document supports a long term goal of NREL to develop technology that identifies the location of a fault in an electrical grid using data from sparse detectors for electro-magnetic traveling waves launched from the fault event. The algorithms in this work were developed to address shortcomings in rational approximation algorithms currently used in electro-magnetic transients (EMT) programs. The industry standard rational approximation algorithm for EMT programs is the Vector Fitting (VF) algorithm. VF has the substantial limitation in that users have no control over the final error of the approximations. The user selects the number of poles, and an initial guess for the location of these poles and VF runs until a convergence criterion is met. There are no general results for the convergence of VF [18] and VF is known to fail to converge for some specific examples [38]. If the solution is unsatisfactory VF must be run again, with more poles, and again there is no guarantee that a satisfactory approximation will be found. The goal of this thesis is to demonstrate an alternative algorithm where the user has control over the final error of the approximation with no initial input from the user about the location or number of poles.

In this work we outline novel algorithms for multi-input multi-output (MIMO) rational approximations. That is, given multiple input functions or function values for multiple functions on a shared sampling set, we construct rational approximations for each function; such that these approximations share a common set of poles. We

compare our algorithm with existing algorithms in terms of accuracy, running time, and number of common poles; each of these aspects of the algorithms is important depending on the target application.

We adapt the well known single-input single-output (SISO) *Adaptive Antoulas–Anderson* (AAA) [35] rational approximation algorithm to MIMO applications. This new multi-input AAA (miAAA) algorithm builds rational approximations in barycentric form and proves to be both fast, robust, and produces approximations with few poles. For some MIMO applications it is also important that all poles of the approximations are stable, that is, all poles have non positive real part. Inspired by other algorithms, we introduce the enforcement of this stability constraint at each iteration to develop the stable multi-input AAA (smiAAA) algorithm. This work demonstrates that the smiAAA algorithm is fast and robust.

To improve the accuracy and to more uniformly bound the errors of our approximations, we describe an algorithm extending the barycentric Lawson-type optimization found in the rational mini-max algorithm [14] to our MIMO approximations. For those applications where the stability of the poles is required, we combine the elements of Lawson optimization with that of stability enforcement.

Combining these tools, we construct a symmetric smiAAA algorithm for model order reduction problems like the EMT problem mentioned previously. The smiAAA algorithm is compared with the current industry standard algorithm, Vector Fitting, as well as the more recently published RKFIT algorithm, which has been suggested for these problems. This work demonstrates that the symmetric smiAAA algorithm often produces better results in accuracy than VF while RKFIT regularly outperforms both. The most important distinction of the symmetric smiAAA algorithm is the user defined target error, while in both VF and RKFIT, the user simply selects the number of initial parameters providing no control over the final error. Therefore, if

the final error is not satisfactory, both VF and RKFIT must be rerun with additional parameters.

In sum, the algorithms developed for this thesis work provide a fast and robust computation of accurate, near mini-max MIMO rational approximations with the *unique feature of user controlled final error*. Initial results from this work were published in [31] and the poster presented at IEEE PES 2020 won 1st place in the student poster competition.

### 1.0.1 Benchmark Example: EMT

The following example problem from EMT serves to motivate the problem of rational approximation for the reader, and it is used as the basic example to demonstrate the performance of each algorithm in this work. We use six functions sampled at 300 points presented in [11, 19, 32, 33]. These entries are all complex valued, and their magnitudes are plotted in Figure 1.1. These functions come from an admittance matrix for the power network in Figure 1.2 from [19]. The graphs of these functions have many sharp *corners*, and in some regions they change very abruptly. Polynomial approximations would therefore be a poor choice for representing these functions. In contrast, rational functions, can easily capture this behavior. In the chapters that follow we use this example to demonstrate the capabilities and contrast the differences in results using miAAA and smiAAA, both before and after Lawson optimization.

### 1.0.2 Summary

The fundamental forms of rational approximation are summarized in Chapter 2. This includes barycentric and partial fraction representations of rational functions and motivating applications of MIMO rational approximations. In Chapter 3, the RKFIT and VF, algorithms with which we compare, are stated carefully so that we may

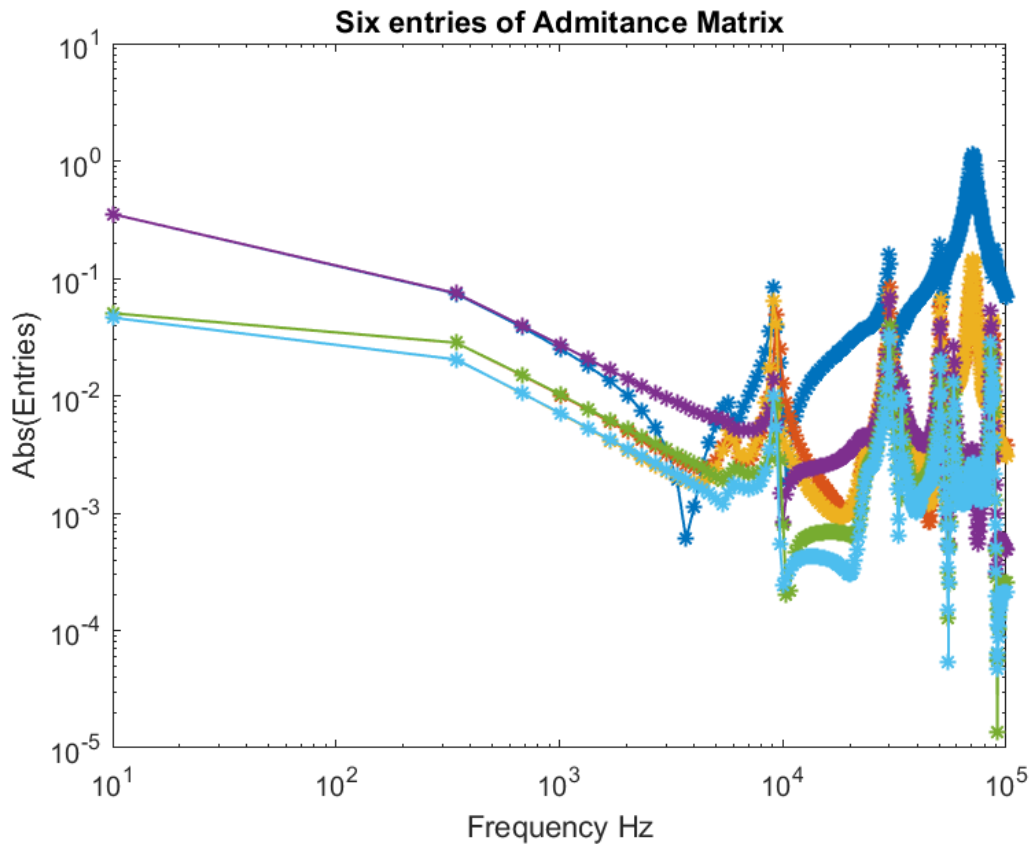


Figure 1.1: Logarithmic absolute value of the 6 entries from the admittance matrix

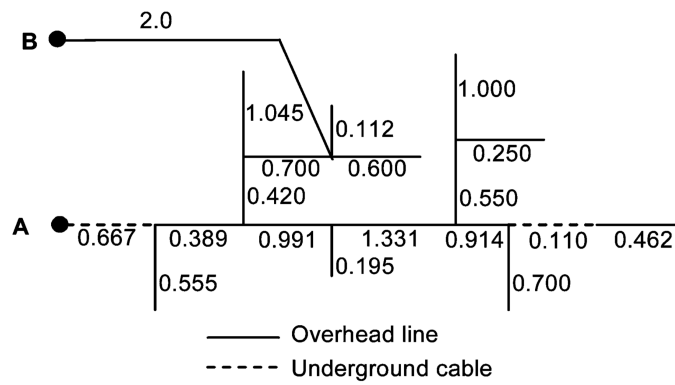


Figure 1.2: Schematic of MIMO power network from [19], with cable lengths given in km

compare and contrast methodologies. The AAA and barycentric Lawson optimization algorithms are discussed in detail, as they are the foundation of the algorithms described in this work. The MIMO extension of AAA, presented in Chapter 4, allows for MIMO rational approximation with the desirable characteristics of the AAA algorithm, including user controlled final error with no initial input about the number or location of the poles. In Chapter 5, we detail the smiAAA algorithm, which maintains the desirable properties of the miAAA algorithm while guaranteeing that all poles of the approximation are stable. The miAAA and smiAAA approximations are used to initialize the barycentric Lawson optimization algorithms in Chapter 6. The resulting approximations have reduced and more uniformly bounded errors. The conversion from barycentric representation to partial fraction representation is important in some applications and is discussed in Chapter 7, with special attention paid to the prevention of spurious poles being computed. Lastly, in Chapter 8, we compare the symmetric smiAAA algorithm with VF and RKFIT on benchmark problems and demonstrate the strengths and weaknesses of the algorithms presented in this work.

## CHAPTER TWO

## RATIONAL APPROXIMATIONS AND THEIR REPRESENTATIONS

The main body of this work concerns itself with Multi input multi output (MIMO) rational approximations. We begin with a brief discussion of single input single output (SISO) rational approximations and the linearization techniques used to construct them. These tools provide the central building blocks of this work.

2.1 Single Input Single Output Rational Approximations

Rational functions have proven to be a flexible and powerful tool in approximation theory.

**Definition 1.** *Given integers  $p > 0$  and  $q > 0$ , we define a rational function of type  $(p, q)$  as*

$$r(z) = \frac{P(z)}{Q(z)} = \frac{\sum_{n=1}^p a_n z^n}{\sum_{n=1}^q b_n z^n}, \quad (2.1)$$

*for some complex numbers  $\{a_n\}$  and  $\{b_n\}$ . These functions are referred to as being of order  $q$ , as the number of poles (the zeroes of the polynomial  $Q$ ) will be a critical parameter.*

Rational functions offer numerous advantages over polynomials as a tool for approximation. For example, a rational function can approximate a bounded function on an unbounded domain. Rational approximations may also converge faster than their polynomial counterparts when approximating a given function near its singularities. They also have much better interpolation properties. For an elegant treatment of the differences between rational and polynomial approximation and some applications where rational approximations are used, see [40](ch. 26).

The quintessential problem of rational approximation is: Given a function  $f$  defined on some subset of  $\mathbb{C}$ , or on a discrete set of sample points  $\mathbf{Z}$ , find a rational approximation  $r$  of type  $(p, q)$  such that  $r \approx f$  on the domain. The accuracy of the approximation can be defined in terms of different norms. For many applications, it is also highly desirable to minimize the number of poles  $q$ . As an example, let us consider the least squares problem, as we will use this type of approximation error later in the thesis. Given a sample set  $\mathbf{Z} = [Z_1, Z_2, \dots, Z_m]$  and values of a function  $f$  on  $\mathbf{Z}$ , find  $r$  of the form (2.1), such that

$$\sum_{z \in \mathbf{Z}} |r(z) - f(z)|^2 \rightarrow \min. \quad (2.2)$$

This equation is nonlinear in the parameters  $\{b_n\}$  and requires the solution of a nonlinear system of equations, which is computationally expensive. A common approach is to linearize equation (2.2) and solve the linear system

$$\sum_{z \in \mathbf{Z}} |P(z) - Q(z)f(z)|^2 \rightarrow \min. \quad (2.3)$$

This linear system is easy to solve; however, it is not guaranteed to have a unique solution. Importantly, a particular solution of (2.3) is not guaranteed to be a good solution to equation (2.2). In fact, solutions to the linear system may have arbitrarily large misfit for the nonlinear system (2.2). Examples with non-unique solutions and large misfits can be found in the introduction to [5]. A specific example is presented in Section 4.1.1, where the linearization used in this work produces unsatisfactory results. The use of these linearizations is one reason that proving any general properties about the convergence of rational approximation algorithms is very difficult and there are few theoretical results in the literature.

## 2.2 Partial Fraction and Barycentric Representations

There are many possible representations of a rational function other than as the quotient of polynomials as in equations (2.1). In this work, we make use of two additional representations, called partial fraction and barycentric representations.

**Definition 2.** *The partial fraction representation of a rational function,  $r(z)$ , is given by*

$$r(z) = \sum_n \frac{\varphi_n}{z - p_n} + \sum_n \alpha_n z^n, \quad (2.4)$$

where the poles  $\{p_n\}$  are the zeros of the polynomial  $Q$  in equation (2.1) and  $\{\varphi_n\}$  are their associated residues. The polynomial term  $\sum_n \alpha_n z^n$  appears when the numerator  $P$  in equation (2.1) is of higher degree than the denominator  $Q$ , specifically

$$\deg\left(\sum_n \alpha_n z^n\right) = \deg(P) - \deg(Q). \quad (2.5)$$

When there is no polynomial term present in equation (2.4), we call  $r(z)$  a proper rational function.

The partial fraction representation is particularly important, as the proper rational function part can easily be Fourier- or Laplace-transformed into a sum of exponential functions, and the transforms of the polynomial parts can be interpreted as derivatives.

The next representation we employ is called the *barycentric representation*.

**Definition 3.** *Given  $N$  distinct points  $z_n$  and samples  $f(z_n)$ , we define the rational function of type  $(p, q)$ , where  $p, q \in \mathbf{Z}^+$  and  $p \leq N - 1$ ,  $q \leq N - 1$ , in barycentric*

form as

$$B(z) = \frac{\sum_{n=1}^N \frac{w_n f(z_n)}{z - z_n}}{\sum_{n=1}^N \frac{w_n}{z - z_n}}, \quad (2.6)$$

where each  $w_n$  is a non-zero barycentric weight, and the distinct points  $z_n$  are referred to as support points. The function  $B(z)$  could have different degrees for its numerator and denominator. In particular,  $B(z)$  may be the sum of a proper rational function and a polynomial as in equation (2.4).

The barycentric formula in equation (2.6) has two important properties:

1. Regardless of the particular choice of barycentric weights  $w_n$ , the function  $B(z)$  in (2.6) interpolates  $f$  at the points  $\{z_m\}$ , that is for  $m = 1, \dots, N$

$$B(z_m) = \lim_{z \rightarrow z_m} \frac{\sum_{n=1}^N \frac{w_n f(z_n)}{z - z_n}}{\sum_{n=1}^N \frac{w_n}{z - z_n}} = \lim_{z \rightarrow z_m} \frac{\frac{w_m f(z_m)}{z - z_m}}{\frac{w_m}{z - z_m}} = \frac{w_m f(z_m)}{w_m} = f(z_m). \quad (2.7)$$

2. The function  $B$  has no poles at the points  $\{z_n\}$ , as  $B(z_n) = f(z_n)$ ,  $n = 1, \dots, N$ .

Conversion between representations is often needed in applications and for comparisons between methods. To convert a barycentric representation  $B$  to a partial fraction representation, we first compute the zeros of the *barycentric denominator* from equation (2.6)

$$D = \sum_{n=1}^N \frac{w_n}{z - z_n} = 0, \quad (2.8)$$

to obtain the poles of  $B$ . These poles, together with the *barycentric numerator*  $J = \sum_{n=1}^N \frac{w_n f(z_n)}{z - z_n}$ , are used to obtain the residues of  $B$  and, if present, polynomial coefficients. Details of this conversion are the topic of Chapter 7.

There exist other barycentric formulas similar to equation (2.6) with slightly different properties. The Vector Fitting algorithm described in Section 3.1.1 uses the slightly

different barycentric formula

$$r(z) = \frac{\sum_{n=1}^N \frac{\varphi_n}{z-p_n}}{1 + \sum_{n=1}^N \frac{\psi_n}{z-p_n}}, \quad (2.9)$$

which has slightly different properties:

1. At the interpolation nodes  $\{p_m\}$ , if  $\psi_m \neq 0$  then for  $m=1, \dots, N$ .

$$r(p_m) = \lim_{z \rightarrow p_m} \frac{\sum_{n=1}^N \frac{\varphi_n}{z-p_n}}{1 + \sum_{n=1}^N \frac{\psi_n}{z-p_n}} = \lim_{z \rightarrow p_m} \frac{\frac{\varphi_m}{z-p_m}}{\frac{\psi_m}{z-p_m}} = \frac{\varphi_m}{\psi_m}. \quad (2.10)$$

2. If  $\forall n, \psi_n = 0$ , then  $r(z)$  is the proper rational function  $r(z) = \sum_n \frac{\varphi_n}{z-p_n}$ .

### 2.3 Multi-Input Multi-Output Rational Approximations

The Multi-input Multi-output (MIMO) rational approximation problem is a natural extension of the single input rational approximation problem.

**Definition 4.** *Given a sample set  $\mathbf{Z}$  and input functions  $\{f_k\}_{k=1}^K$  evaluated on  $\mathbf{Z}$ , find rational approximations*

$$r_k(z) = \frac{P_k(z)}{Q(z)} \approx f_k, \quad k = 1, 2, \dots, K. \quad (2.11)$$

It is important to note that the denominator for each approximation  $r_k$  given above is the same. This is equivalent to the approximations sharing a common set of poles  $\{p_n\}$  with different residues  $\{\varphi_{k,n}\}$  and different polynomial coefficients  $\{\alpha_{k,n}\}$  when written in partial fraction form

$$r_k(z) = \sum_n \frac{\varphi_{k,n}}{z-p_n} + \sum_n \alpha_{k,n} z^n, \quad k = 1, 2, \dots, K. \quad (2.12)$$

The number of common poles  $\{p_n\}$  is a critical parameter in MIMO rational approximation applications, which are discussed in Section 2.5.

## 2.4 Vector Norms

Throughout this work, we will measure the error between approximations and function samples using three different norms. The first vector norm of interest is the discrete  $L_2$ -norm  $\|\cdot\|_2$

$$\|f - b\|_2 := \sqrt{\sum_{z \in \mathbf{Z}} (f(z) - b(z))^2}. \quad (2.13)$$

We use the  $L_2$  error to select some parameters when constructing our approximations; it is an important metric for measuring the accuracy of an approximation for some applications we consider. The second norm we employ is the  $L_\infty$ -norm,  $\|\cdot\|_\infty$

$$\|f - b\|_\infty := \max_{z \in \mathbf{Z}} |f(z) - b(z)|. \quad (2.14)$$

For some applications, this will be the most important measure of the accuracy of the approximations; we also use this norm when selecting certain parameters during the construction of our approximations.

The third norm we consider is the  $\mathcal{H}_2$  norm. For the MIMO rational approximation problem the  $\mathcal{H}_2$  error is given by

$$\|\{f_k\}_{k=1}^K - \{r_k\}_{k=1}^K\|_{\mathcal{H}_2} := \sqrt{\frac{\sum_{k=1}^K \|f_k(z) - r_k(z)\|_2^2}{\sum_{k=1}^K \|f_k(z)\|_2^2}}. \quad (2.15)$$

For certain applications this is the most important measure of accuracy of the approximations.

## 2.5 Applications of MIMO Rational Approximations

Although EMT simulation problems were the initial motivation for this research, we briefly discuss the broader problem of model order reduction for time invariant linear dynamical systems, of which EMT is one example. We also discuss the nonlinear eigenvalue problem, as this work can be used in these applications, and may address some specific difficulties therein.

### 2.5.1 Linear Time-Invariant Dynamical Systems

Model order reduction of linear time invariant dynamical systems (LTI), is a classical problem in control theory [1]. The electro-magnetic transients (EMT) modeling problem, on which this research is based, is an example of a LTI so we use some examples from EMT to illustrate our algorithm in later sections.

**Definition 5.** *A LTI is a system of equations of the form*

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t), \end{cases} \quad (2.16)$$

where  $u(t) \in \mathbb{R}^{m \times 1}$  is the input to the system,  $y(t) \in \mathbb{R}^{p \times 1}$  is the output and  $x(t) \in \mathbb{R}^{N \times 1}$  is the internal variable or state variable. The system has real constant matrices  $A \in \mathbb{R}^{N \times N}$ ,  $B \in \mathbb{R}^{N \times m}$ ,  $C \in \mathbb{R}^{p \times N}$ , and  $D \in \mathbb{R}^{p \times m}$ . If  $m = p = 1$  the system is SISO; if  $m \geq 1$ , the system has multiple inputs, and if  $p \geq 1$  the system has multiple outputs.

Taking a Laplace transform, we can rewrite the system in (2.16) as

$$\tilde{y}(s) = [C(sI - A)^{-1}B + D]\tilde{u}(s), \quad (2.17)$$

where  $\tilde{y}(s)$  and  $\tilde{u}(s)$  are the Laplace transforms of  $y(t)$  and  $u(t)$ , respectively, and  $I$  is the  $N \times N$  identity matrix. The advantage of taking the Laplace transform is that one can now directly compute  $\tilde{y}(s)$  from  $\tilde{u}(s)$  by multiplication with the, so called in control theory [6] (p.50), transfer function

$$G(s) = C(sI - A)^{-1}B + D. \quad (2.18)$$

When the number of state variables  $N$  is large, simulation of the system can be very computationally expensive. Therefore one seeks a reduced order model

$$\begin{cases} \dot{x}_r(t) = A_r x_r(t) + B_r u(t) \\ y_r(t) = C_r x_r(t) + D_r u(t), \end{cases} \quad (2.19)$$

where  $A_r \in \mathbb{R}^{n \times n}$ ,  $B_r \in \mathbb{R}^{n \times m}$ ,  $C_r \in \mathbb{R}^{p \times n}$  and  $D_r \in \mathbb{R}^{p \times m}$ , with  $n \ll N$  such that the reduced outputs are a good approximation of the corresponding true outputs from the full model. The reduced order problem has a reduced order transfer function:

$$G_r(s) = C_r(sI_r - A_r)^{-1}B_r + D_r. \quad (2.20)$$

Rational functions with a common set of poles are used to approximate the rows of the transfer function  $G(s)$  in order to find a reduced  $G_r(s)$ . The order of the rational approximations is the order of the reduced model, since the poles of the rational approximations are the singularities of  $sI_r - A_r$  in the reduced order model. The most important error measure for these rational approximations in LTI is the  $\mathcal{H}_2$ -error as, minimization of  $\mathcal{H}_2$  error in the approximation of the transfer function

corresponds to minimization of the  $L_\infty$  error of the time domain solution [16] as

$$\|y(t) - y_r(t)\|_\infty \leq \|G - G_r\|_{\mathcal{H}_2} \|u(t)\|_2. \quad (2.21)$$

### 2.5.2 Nonlinear Eigenvalue Problems

The nonlinear eigenvalue problem (NEP) [22] is another application, where MIMO rational approximations are used. The set-valued AAA algorithm [28] and the weighted set-valued AAA algorithm [39], with which we compare later, were developed specifically for approximating NEP's. The NEP is defined as follows:

**Definition 6.** *Given a non-empty open set  $\Omega \subseteq \mathbb{C}$  and matrix valued function  $F: \Omega \mapsto \mathbb{C}^{n \times n}$ , find the scalars  $\lambda \in \Omega$  and nonzero right and left eigenvectors,  $\mathbf{v}, \mathbf{w} \in \mathbb{C}^n$ , such that;*

$$F(\lambda)\mathbf{v} = \mathbf{0} \text{ and } \mathbf{w}^*F(\lambda) = \mathbf{0}, \quad (2.22)$$

where  $\mathbf{w}^*$  is the conjugate transpose of  $\mathbf{w}$  and  $\mathbf{0}$  denotes the zero column vector of length  $n$ .

The dependence of  $F(z)$  on  $z$  is typically nonlinear while the solutions  $\lambda$  to equation (2.22) solve the scalar equation  $\det(F(z)) = 0$ ; hence the name *nonlinear eigenvalue problem*. To solve an NEP with MIMO rational approximations it must be formulated in a split form:

$$F(z) = \sum_k f_k(z)A_k, \quad (2.23)$$

where the constant matrices  $A_k \in \mathbb{C}^{n \times n}$  are called the coefficient matrices, and the complex valued functions  $f_k: \Omega \mapsto \mathbb{C}$ . If the  $z$ -dependence of  $F(z)$  is nonlinear then the functions  $\{f_k\}$  will be nonlinear. The nonlinear functions  $\{f_k\}$  are approximated with MIMO rational approximations. Although these approximations are nonlinear,

they are highly structured, all being rational functions sharing the same set of poles. Powerful linearization techniques exist [28] for using the MIMO rational approximations of  $\{f_k\}$ , and the coefficient matrices  $A_k$ , to build linear systems whose eigenvalues and eigenvectors approximate the solutions  $\lambda, \mathbf{w}, \mathbf{v}$  of equation (2.22). The number of common poles used in the approximations is a factor in the size of the final linearized systems. The most important error measurement for the rational approximations in NEPs is the  $L_\infty$  error [39].

## CHAPTER THREE

## CURRENT ALGORITHMS

In this chapter, four algorithms are presented. The first two algorithms, Vector Fitting and RKFIT, are MIMO rational approximation algorithms. We will compare the results of these algorithms with the symmetric smiAAA algorithm in Chapter 8. The other two algorithms, AAA and barycentric Lawson optimization, are a SISO rational approximation algorithm and a post processing algorithm for SISO approximations in barycentric form respectively. The miAAA, smiAAA, and symmetric smiAAA algorithms are extensions of these algorithms to the MIMO problem.

### 3.1 Current Algorithms for MIMO Rational Approximation

Currently there are two algorithms for constructing MIMO rational approximations that have been used or suggested for EMT problems. The Vector Fitting Algorithm [21] was developed in the 1970s and it is widely used in EMT simulation software today. The very recent RKFIT algorithm was first published in 2017 and distributed as part of the Rational Krylov toolbox for MATLAB. It has also been suggested for EMT problems [34]. The goal of this thesis is to develop a third algorithm, called symmetric smiAAA, and to present a comparison of the symmetric smiAAA algorithm with these two existing algorithms. The three algorithms are evaluated based on accuracy of approximation, order of approximation, and running time. A brief description of these algorithms is provided so that we may compare and contrast the three methodologies. Additional algorithms for LTI systems which have not been directly applied to EMT models can be found in [2], [26], [16]. We note that

these algorithms also require initial guessing for the number and location of the poles similar to VF and RKFIt but we will not perform any comparison with them.

### 3.1.1 The Vector Fitting Algorithm

The Vector Fitting (VF) algorithm is detailed in [12], [18], and [21]. It is an iterative algorithm that seeks rational functions  $r_k$  approximating the functions  $f_k$  on the sample set  $\{Z_n\}_{n=1}^M$ . A brief overview of the essential elements of VF is presented next.

For approximation of a single function, given an initial guess for the location of the poles  $\{p_n\}_{n=1}^N$ , VF seeks to improve the poles as follows. Let  $r$  be of the barycentric form

$$r(z) = \frac{\sum_{n=1}^N \frac{\varphi_n}{z-p_n}}{1 + \sum_{n=1}^N \frac{\psi_n}{z-p_n}}, \quad (3.1)$$

with interpolation nodes  $\{p_n\}$ , and unknown residues  $\boldsymbol{\varphi} = [\varphi_1, \varphi_2, \dots, \varphi_N]^T$  and  $\boldsymbol{\psi} = [\psi_1, \psi_2, \dots, \psi_N]^T$ . Linearizing equation (3.1) we arrive at

$$r(z) \left( 1 + \sum_{n=1}^N \frac{\psi_n}{z-p_n} \right) = \sum_{n=1}^N \frac{\varphi_n}{z-p_n}. \quad (3.2)$$

This equation is solved in a least squares sense over the samples  $Z$  with the linear system

$$\begin{bmatrix} \frac{1}{Z_1-p_1} & \cdots & \frac{1}{Z_1-p_N} & \frac{-f(Z_1)}{Z_1-p_1} & \cdots & \frac{-f(Z_1)}{Z_1-p_N} \\ \vdots & & \vdots & \vdots & & \vdots \\ \frac{1}{Z_M-p_1} & \cdots & \frac{1}{Z_M-p_N} & \frac{-f(Z_M)}{Z_M-p_1} & \cdots & \frac{-f(Z_M)}{Z_M-p_N} \end{bmatrix} \begin{bmatrix} \boldsymbol{\varphi} \\ \boldsymbol{\psi} \end{bmatrix} = \begin{bmatrix} f(Z_1) \\ \vdots \\ f(Z_M) \end{bmatrix}. \quad (3.3)$$

The poles  $\{p_n\}$  are then replaced with the roots of the denominator  $1 + \sum_{n=1}^N \frac{\psi_n}{z-p_n}$ , which are found by computing the eigenvalues of the matrix  $\text{diag}(p_n) - \boldsymbol{\psi} \mathbf{1}^T$ , where  $\mathbf{1}^T = [1 \ 1 \ \dots \ 1]^T \in \mathbb{R}^N$ , see [15] (Chapter 9). Iteration continues until  $\|\boldsymbol{\psi}\|_2$  is very

small, that is the denominator of  $r(z)$  in (3.1) can be considered to be 1. The final approximation is given by

$$r_k(z) = \sum_{n=1}^N \frac{\varphi_n}{z - p_n}. \quad (3.4)$$

according to property 2 of equation (3.1). To solve the MIMO rational approximation problem with VF, equation (3.5) is solved for all functions  $\{f_k\}_{k=1}^K$  simultaneously, with the same  $\psi$  and a different  $\varphi_k$  for each function. For example, for  $K = 2$  input functions, the least squares problem is given by  $A\mathbf{x} = \mathbf{b}$  where

$$A = \begin{bmatrix} \frac{1}{Z_1-p_1} & \cdots & \frac{1}{Z_1-p_N} & & & \frac{-f_1(Z_1)}{Z_1-p_1} & \cdots & \frac{-f_1(Z_1)}{Z_1-p_N} \\ \vdots & & \vdots & 0 & & \vdots & & \\ \frac{1}{Z_M-p_1} & \cdots & \frac{1}{Z_M-p_N} & & & \frac{-f_1(Z_M)}{Z_M-p_1} & \cdots & \frac{-f_1(Z_M)}{Z_M-p_N} \\ & & & \frac{1}{Z_1-p_1} & \cdots & \frac{1}{Z_1-p_N} & \frac{-f_2(Z_1)}{Z_1-p_1} & \cdots & \frac{-f_2(Z_1)}{Z_1-p_N} \\ & 0 & & \vdots & & \vdots & \vdots & & \vdots \\ & & & \frac{1}{Z_M-p_1} & \cdots & \frac{1}{Z_M-p_N} & \frac{-f_2(Z_M)}{Z_M-p_1} & \cdots & \frac{-f_2(Z_M)}{Z_M-p_N} \end{bmatrix} \quad (3.5)$$

$$\mathbf{x} = [\varphi_1, \varphi_2, \psi]^T, \text{ and } \mathbf{b} = [f_1(Z_1), \dots, f_1(Z_M), f_2(Z_1), \dots, f_2(Z_M)]^T.$$

In practice, the sparsity of this system is exploited for a more efficient solution. Analysis of the convergence of VF is detailed in [27] and [38]. The MATLAB implementations of VF used for comparison in this thesis are available from “The Vector Fitting Website” <https://www.sintef.no/projectweb/vectorfitting/> in the VFIT3 and Matrix Fitting Toolbox (MFT) packages for MATLAB.

### 3.1.2 RKFIT

The recently developed algorithm RKFIT [3–5], can also be used to solve the MIMO rational approximation problem. RKFIT is also an iterative algorithm that seeks rational functions  $r_k$  approximating the functions  $f_k$  on the sample set  $\mathbf{Z} =$

$\{Z_n\}_{n=1}^M$ . RKFIT explicitly seeks approximations that minimize the  $H_2$  error.

The RKFIT algorithm was developed to solve a more general matrix-valued rational approximation problem. An attempt at a brief description is given here. Given an interpolation node matrix  $A$ , a family of data matrices  $\{F^{[j]}\}_{j=1}^\ell$  with elementwise weight matrices  $D^{[j]}$ , and a block matrix of spectral weight vectors  $B$ , RKFIT seeks rational approximations  $r^{[j]}(A)$  which minimize

$$\sqrt{\frac{\sum_{j=1}^\ell \|D^{[j]}[F^{[j]}B - r^{[j]}(A)B]\|_F^2}{\sum_{j=1}^\ell \|D^{[j]}F^{[j]}B\|_F^2}}. \quad (3.6)$$

The MIMO rational approximation problem we are considering is a special case of this matrix-valued problem where  $A = \text{diag}(\mathbf{Z})$ ,  $F^{[j]} = \text{diag}(f_j)$  and the spectral and element wise weights  $B$  and  $D^{[j]}$  are all ones.

The rational approximations are constructed in the RKFUN representation which takes the form of a triple  $r_d = (H_d, K_d, c)$  where  $c$  is a coefficient vector and  $H_d, K_d \in \mathbb{C}^{(d+1) \times d}$  are upper-Hessenberg matrices satisfying a rational Arnoldi decomposition  $AV_{d+1}K_d = V_{d+1}H_d$  associated with a rational Krylov space

$$\mathbb{Q}_{d+1}(A, b) = q(A)^{-1} \text{span}\{b, Ab, \dots, A^d b\}, \quad (3.7)$$

where  $A \in \mathbb{C}^{l \times l}$ ,  $b \in \mathbb{C}^l$ ,  $V_{d+1} \in \mathbb{C}^{l \times (d+1)}$ , and  $q \in \mathbb{P}_d$  is a monic polynomial such that  $q(A)$  is invertible. The RKFUN representation can be converted to a barycentric representation as described in [13]. The Arnoldi decomposition encodes a basis of rational functions  $r_k = p_k/q$ , with  $p_k \in \mathbb{P}_d$ , all sharing the same denominator  $q$  whose roots, the poles of  $r_k$ , are given by  $\psi_j = h_{j+1,j}/k_{j+1,j}$ , the ratio of the sub-diagonal elements of  $H_d$  and  $K_d$ . For any  $c \in \mathbb{C}$ ,  $r^{[j]}(z)$  is evaluated by  $r^{[j]}(z) = n(z) \cdot c$ , where  $n(z)$  is the (unique) left null vector of  $zK^{[j]} - H^{[j]}$  normalized with first component

equal to one. Beginning with a decomposition  $AV_{d+1}K_d = V_{d+1}H_d$  with poles  $\psi_1, \psi_2, \dots$ , each iteration of RKFIT solves the least squares problem

$$\hat{v} = \operatorname{argmin} \|(I - V_{d+1}V_{d+1}^*)FV_{d+1}c\|_2, \quad \text{s.t.} \quad \|c\|_2 = 1. \quad (3.8)$$

The poles of  $\hat{r}$  associated with  $(\hat{v}) = (\hat{r})(A)b$  are used as the initial poles of the next iteration.

Theorem 2.2 in [5] proves that: Assuming computation with perfect arithmetic, if the input samples being approximated come from a truly rational matrix-valued function, and if the iteration begins with as many or more poles as the underlying matrix-valued function, then this iteration will converge to the exact poles (plus potential extraneous poles) in a single iteration.

In practice this pole relocation process is iterated until the poles satisfy a convergence criterion. After convergence, under the same assumptions, Lemma 2.1 in [5] shows that the number of extraneous poles can be found by computing the defect of

$$S = (I - P_T)FV_{M+1}. \quad (3.9)$$

After reducing the number of poles by the defect of  $S$ , the pole relocation can be performed again until convergence. These two steps are the theoretical underpinning of the RKFIT algorithm. The RKtoolbox for MATLAB used for comparison in this thesis is available at <http://guettel.com/rktoolbox/>.

### 3.2 The AAA Algorithm and Barycentric Lawson Optimization

This thesis extends the AAA algorithm [35] and barycentric Lawson optimization [14] to the MIMO rational approximation problem. These algorithms are implemented

in MATLAB as part of the open source package *Chebfun* (see [www.chebfun.org](http://www.chebfun.org)). We introduce these two algorithms here, as they are the foundation of the algorithms presented in later chapters.

### 3.2.1 AAA Algorithm

The AAA algorithm constructs SISO rational approximations as follows. Given values of the function  $f$  at sample points  $\mathbf{Z} = \{Z_m\}_{m=1}^M$ , AAA builds rational approximations in the barycentric form in equation (2.6). Specifically it constructs

$$B(z) = \frac{\sum_{n=1}^N \frac{w_n f(z_n)}{z - z_n}}{\sum_{n=1}^N \frac{w_n}{z - z_n}}, \quad (3.10)$$

where  $z_n$  are called the support points,  $w_n$  are called the barycentric weights and the barycentric formula has the noteworthy property that if all the weights  $w_n \neq 0$ , then  $B$  perfectly interpolates the function  $f$  at the support points  $z_n$ .

AAA chooses support points from the sampling set  $\mathbf{Z}$  iteratively based on the criterion of being the sample point where the current barycentric approximation has the largest deviation from  $f$ . This iterative process is outlined here: The first support point is chosen as the sample with the largest deviation from

$$\text{mean}(f) = \frac{\sum_{z \in \mathbf{Z}} f(z)}{M}, \quad (3.11)$$

that is

$$z_1 = \arg \max_{z \in \mathbf{Z}} |f(z) - \text{mean}(f)|. \quad (3.12)$$

After iteration  $i$ , we have  $i$  support points, and we denote the set of those sample points not chosen as support points as  $\mathbf{Z}^{(i)}$ . In order to complete the calculation of the barycentric form, it remains to determine the weights. The barycentric weights

in equation (3.16) are found at each iteration by solving the linearized system

$$\min_{z \in \mathbf{Z}^{(i)}} \|J(z) - f(z)D(z)\|_2, \quad s.t. \quad \|\mathbf{w}\|_2 = 1. \quad (3.13)$$

where  $\mathbf{w}$  is the vector containing the barycentric weights. This linear system is not guaranteed to produce a good solution to the non-linear problem  $B(z) \approx f$ , however it is easy to solve. For clarity we rewrite the system as the least squares problem

$$L\mathbf{w} = 0, \quad s.t. \quad \|\mathbf{w}\|_2 = 1, \quad (3.14)$$

Here,  $L$  is the Löwner matrix [30] with entries

$$L_{h,j} = \frac{f(Z_h^{(N)}) - f(z_j)}{Z_h^{(N)} - z_j}. \quad (3.15)$$

We now have the intermediate barycentric representation

$$B^{(i)}(z) = \frac{\sum_{n=1}^i \frac{w_n f(z_n)}{z - z_n}}{\sum_{n=1}^i \frac{w_n}{z - z_n}}, \quad (3.16)$$

We then compute the error  $\|f - B^{(i)}\|_\infty$  and if the desired tolerance is not reached, we continue by adding another support point. The structure of  $L$  can be exploited by defining the Cauchy matrix [9],  $C \in \mathbb{C}^{(M-i) \times i}$

$$C = \begin{bmatrix} \frac{1}{Z_1 - z_1} & \frac{1}{Z_1 - z_2} & \cdots & \frac{1}{Z_1 - z_i} \\ \frac{1}{Z_2 - z_1} & \frac{1}{Z_2 - z_2} & \cdots & \vdots \\ \vdots & & \vdots & \\ \frac{1}{Z_{(M-i)} - z_1} & \frac{1}{Z_{(M-i)} - z_2} & \cdots & \frac{1}{Z_{(M-i)} - z_i} \end{bmatrix}, \quad (3.17)$$

and rewriting the matrix  $L$  as follows

$$L = \begin{bmatrix} f(Z_1^{(i)}) & & & \\ & f(Z_2^{(i)}) & & \\ & & \ddots & \\ & & & f(Z_{M-i}^{(i)}) \end{bmatrix} C - C \begin{bmatrix} f(z_1) & & & \\ & f(z_2) & & \\ & & \ddots & \\ & & & f(z_i) \end{bmatrix} \quad (3.18)$$

or, equivalently,

$$L = \text{diag}(f(\mathbf{Z}^{(i)}))C - C\text{diag}(f(\mathbf{z})), \quad (3.19)$$

where  $\mathbf{z} = [z_1, z_2, \dots, z_i]$ . In practice, to gain efficiency in computation, we update the successive matrices  $C$  each time a new support point is chosen by first adding an additional column to  $C$  and then removing the row corresponding to the new support point. Although these Cauchy matrices can be ill-conditioned this linearization has proven very effective in nearly every problem we have run. We provide a single example where this linearization fails in Section 4.1.1. This example is intended to illustrate the risk of this linearization to return poor results. Pseudocode for the AAA algorithm is provided in Algorithm 3.1.

### 3.2.2 Lawson Optimization of Barycentric Interpolants

As AAA employs a linearization of a nonlinear problem and a greedy selection of parameters it does not guarantee optimality in any norm. For applications where the  $L_\infty$  error is most important, a SISO post processing optimization algorithm for barycentric interpolants is introduced, in [14]. This algorithm is based on Lawson's iterative re-weighted least squares method [37]. We describe it here, and we extend it to MIMO problems with and without stability constraints in Chapter 6. We define

---

Algorithm 3.1: AAA

---

- 1: Set barycentric interpolant  $B = \text{mean}(f)$
  - 2: **while** error > tolerance **do**
  - 3: Select the next support point  $z_i = \text{argmax}_{z \in \mathbf{Z}^{(i)}} |B(z) - f(z)|$
  - 4: Compute the Barycentric interpolant with linear least squares  
 $L\mathbf{w} = 0, \|\mathbf{w}\|_2 = 1$
  - 5: Compute the error  $\|B^{(i)} - f\|_\infty$
  - 6: **end while**
  - 7: **return** barycentric interpolant
- 

a non-interpolatory barycentric formula  $B^\wedge$  related to equation (2.6).

$$B^\wedge(z) = \frac{J^\wedge}{D^\vee} = \frac{\sum_{n=1}^N \frac{w_n^\wedge f(z_n)}{z - z_n}}{\sum_{n=1}^N \frac{w_n^\vee}{z - z_n}}. \quad (3.20)$$

The weights in the numerator,  $\mathbf{w}^\wedge$ , and the weights in the denominator,  $\mathbf{w}^\vee$ , of this non-interpolatory barycentric formula can be different. Introducing these additional degrees of freedom allows us to improve the approximations over the sample points not chosen as support points. However, there are drawbacks. The modified barycentric form  $B^\wedge$  no longer perfectly interpolates  $f$  at the support points  $\{z_n\}$  and instead  $B^\wedge(z_n) = w_n^\wedge f(z_n)/w_n^\vee$ . The weights  $\mathbf{w}^\wedge$  and  $\mathbf{w}^\vee$  are chosen as follows. The single input Lawson optimization begins with a vector of Lawson weights  $\mathbf{l}\mathbf{w} \in \mathbb{R}^M$ , initially with all components set to one. The algorithm first solves the weighted linearized system

$$\min_{z \in \mathbf{Z}} \left\| \sqrt{\mathbf{l}\mathbf{w}} [J^\wedge(z) - f(z)D^\vee(z)] \right\|_2, \quad \|\mathbf{w}^\wedge\|_2 + \|\mathbf{w}^\vee\|_2 = 1. \quad (3.21)$$

The matrix  $[J^\wedge(z) \quad -f(z)D^\vee(z)]$ , although similar to that in equation 3.13, has the distinction that the blocks  $[J^\wedge]$  and  $[-f(z)D^\vee(z)]$  are now separate, as they no longer share common weights. Equation (3.21) is solved using the SVD of the following weighted non-interpolatory Löwner type matrix.

$$\text{diag}(\sqrt{\mathbf{l}\mathbf{w}})[\text{diag}(f(\mathbf{Z}))C^l \quad -C^l\text{diag}(f(\mathbf{z}))]. \quad (3.22)$$

Here  $\mathbf{z} = [z_1, z_2, \dots, z_N]$  and  $C^l$  denotes the modified Cauchy matrix with entries

$$C_{i,j}^l = \frac{1}{Z_i - z_j}, \quad i = 1, 2, \dots, M, \quad j = 1, 2, \dots, N, \quad (3.23)$$

and with the rows of the matrix where  $Z_i$  was chosen as a support point replaced with sparse rows containing a single 1 and a single -1. These rows minimize  $w_i^\wedge - w_i^\vee$  in equation (3.21) to maintain a good approximation at the support points. The minimization is performed over all sample points including those chosen as support points. The Lawson weights are then updated by

$$lw'_m = lw_m \cdot |B^\wedge(Z_m) - f(Z_m)|, \quad Z_m \in Z \quad (3.24)$$

and normalized so  $\|\mathbf{l}\mathbf{w}'\|_2 = 1$ . We then return to equation (3.21) with the new Lawson weights  $\mathbf{l}\mathbf{w}'$ .

The iteration continues until  $\|\mathbf{l}\mathbf{w}' - \mathbf{l}\mathbf{w}\|_2 < tol$  between two consecutive iterations, or we reach a predefined max number of iterations. Any iteration of the Lawson algorithm may produce a worse barycentric approximation due to the linearization, so we save the best approximation found by any iteration (it may not be, and is often not, the final iteration). Pseudocode for single input barycentric Lawson optimization is given in Algorithm 3.2.

---

Algorithm 3.2: Barycentric Lawson Optimization

---

- 1: Initialize Lawson weights  $\mathbf{lw} = [1, 1, \dots, 1]$
  - 2: **for** Max iterations **do**
  - 3: Compute weighted approximation  $\min_{z \in Z} \|\sqrt{\mathbf{lw}}[J(z) - f(z)D(z)]\|_2$   
via the SVD of  $\text{diag}(\sqrt{\mathbf{lw}})[\text{diag}(f(\mathbf{Z}))C^l - C^l\text{diag}(f(\mathbf{z}))]$
  - 4: Compute deviation  $\|B(z) - f(z)\|_\infty$ ,  $z \in \mathbf{Z}$ , and compare with current best approximation
  - 5: Update Lawson weights  $lw'_m = lw_m \cdot |B(Z_m) - f(Z_m)|$ ,  $Z_m \in \mathbf{Z}$
  - 6: **if**  $\|\mathbf{lw}' - \mathbf{lw}\|_2 < tol$  **return** best approximation
  - 7: **end for**
  - 8: **return** best approximation
-

## CHAPTER FOUR

## DEVELOPMENT OF MULTI-INPUT MULTI-OUTPUT AAA

4.1 Multi-Input Multi-Output Extension of AAA

One of the main contributions of this research is the development of a methodology for extending the AAA algorithm in Section 3.2.1 to MIMO problems. We call this MIMO AAA algorithm miAAA. Although similar methodologies were published in [23] and [28] in 2017 and 2018, respectively, the development of the algorithm in this section was carried out independently by the author and prior to the knowledge of the existence of those bearing close similarity to it. Similarities and differences between this work and these papers are discussed in Sections 4.1.3 and 4.1.5. First we outline the smiAAA methodology.

Starting with a sample set  $\mathbf{Z} = \{Z_m\}_{m=1}^M \subseteq \mathbb{C}$  and sample values of  $K$  functions  $\{f_k\}_{k=1}^K$  on the set  $\mathbf{Z}$ , we iteratively build barycentric approximations following the ideas in Section 3.2.1. At the end of the iteration, we obtain a common set of  $N$  support points  $\{z_n\}_{n=1}^N$  and weights  $w_n$  so that the barycentric functions

$$B_k(z) = \frac{\sum_{n=1}^N \frac{w_n f_k(z_n)}{z - z_n}}{\sum_{n=1}^N \frac{w_n}{z - z_n}} \approx f_k(z), \quad k = 1, \dots, K, \quad (4.1)$$

approximate the functions  $f_k$ ,  $k = 1, \dots, K$  within some target accuracy. Since all  $B_k$  have a common barycentric denominator, the final rational approximations, written in partial fraction form, have a common set of poles with different residues and different polynomial parts.

At step  $i$  of the iteration, a greedy choice described below is used to select an

additional support point  $z_i$  from the sample set  $\mathbf{Z}$ . We then solve a least squares problem to compute a set of common weights  $\mathbf{w} = [w_1, \dots, w_i]$  for the intermediate barycentric approximations

$$B_k^{(i)}(z) = \frac{J_k^{(i)}(z)}{D^{(i)}(z)} = \frac{\sum_{n=1}^i \frac{w_n f_k(z_n)}{z - z_n}}{\sum_{n=1}^i \frac{w_n}{z - z_n}}, \quad (4.2)$$

and compute the error  $\|B_k^{(i)} - f_k\|_\infty$  on  $\mathbf{Z}^{(i)} = \mathbf{Z} \setminus \{z_1, \dots, z_i\}$  to pick the next support point.

For the initial step of the algorithm, the first support point  $z_1$  is chosen so that

$$z_1 = \arg \max_{z \in \mathbf{Z}} \left( \sum_k |f_k(z) - \text{mean}(f_k(z))| \right), \quad (4.3)$$

and successive support points  $z_i$  are chosen from  $\mathbf{Z}^{(i)}$  such that

$$z_i = \arg \max_{z \in \mathbf{Z}^{(i)}} \left( \sum_k |f_k(z) - B_k(z)| \right). \quad (4.4)$$

At each iteration we select  $z_i$  by choosing the sample point from  $\mathbf{Z}^{(i)}$  with the largest *common* error for all the functions. Each barycentric approximation  $B_k$  then interpolates  $f_k$  at  $z_i$  removing the largest *common* error from the approximation at iteration  $i - 1$ . If multiple points  $z \in \mathbf{Z}^{(i)}$  maximize equation (4.4) then we simply choose the one with the lowest index.

To determine the set of common weights at step  $i$ , we seek approximations

$$B_k^{(i)}(z) = \frac{J_k^{(i)}(z)}{D^{(i)}(z)}, \quad (4.5)$$

which is a nonlinear system, so we minimize the linearized equation with respect to

$$\mathbf{w} = [w_1, w_2, \dots, w_N]^T$$

$$\min_{1 \leq k \leq K, z \in \mathbf{Z}^{(N)}} \left\| J_k^{(i)}(z) - f_k(z) D^{(i)}(z) \right\|_2, \quad \text{s.t.} \quad \|\mathbf{w}\|_2 = 1. \quad (4.6)$$

We terminate the iteration when  $\max_k |B_k^{(i)}(z) - f_k(z)|_{z \in \mathbf{Z}}$  is less than the desired tolerance; consequently, all the rational approximations are accurate with respect to the vector norm  $\|\cdot\|_\infty$  over the sample points.

To solve equation (4.6) we find the least squares solution  $\mathbf{w}$  to the problem

$$L\mathbf{w} = 0, \quad \|\mathbf{w}\|_2 = 1, \quad \text{with } L = \begin{bmatrix} L_1 \\ \vdots \\ L_K \end{bmatrix}. \quad (4.7)$$

where  $L_k$  is the Löwner matrix as used in AAA for the function  $f_k$

$$L_{k(h,j)} = \frac{f_k(Z_h^{(i)}) - f_k(z_j)}{Z_h^{(i)} - z_j}. \quad (4.8)$$

The structure of  $L_k$  can be exploited in the same way as in equation (3.19) with

$$L_k = \text{diag}(f_k(\mathbf{Z}^{(i)}))C - C\text{diag}(f_k(\mathbf{z})). \quad (4.9)$$

For efficient computation the successive matrices  $C$  can be built by adding a column and removing a row as described in Section 3.2.1. Pseudocode for miAAA is provided in Algorithm 4.3.

#### 4.1.1 Example Failure of Linearization

Inspired by the example in [7] page 108, we discuss a SISO example where the linearization in equation (4.6) may produce poor results. Let the samples of  $f$  be

---

Algorithm 4.3: miAAA

---

- 1: Set barycentric interpolants  $B_k = \text{mean}(f_k)$
  - 2: **while** deviation  $\geq$  tolerance **do**
  - 3: Select the next support point  $z_i = \arg \max_{z \in \mathbf{Z}^{(i)}} (\sum_k |f_k(z) - B_k(z)|)$
  - 4: Compute the Barycentric interpolant with linear least squares  
 $L\mathbf{w} = 0, \|\mathbf{w}\|_2 = 1$
  - 5: Compute deviation  $\max_k \|B_k(z) - f_k(z)\|_\infty$
  - 6: **end while**
  - 7: **return** barycentric interpolant
- 

defined as

$$f_i = \begin{cases} 1 & \text{if } i = 1 \\ 0 & \text{otherwise,} \end{cases} \quad (4.10)$$

and for illustration, let there be five samples  $\mathbf{Z} = \{Z_1, Z_2, \dots, Z_5\}$ . The first support point chosen is  $z_1 = Z_1$  and the linearized least squares problem is

$$L\mathbf{w} = \begin{bmatrix} \frac{1}{Z_2 - z_1} \\ \frac{1}{Z_3 - z_1} \\ \frac{1}{Z_4 - z_1} \\ \frac{1}{Z_5 - z_1} \end{bmatrix} \begin{bmatrix} w_1 \end{bmatrix} = 0, \quad \|\mathbf{w}\|_2 = 1, \quad (4.11)$$

with solution  $w_1 = 1$ . The Barycentric approximation is now the constant function  $B^{(1)}(z) = 1$ . The second support point chosen will be  $z_2 = Z_2$  and the linearized least

squares problem is

$$L\mathbf{w} = \begin{bmatrix} \frac{1}{Z_3-z_1} & 0 \\ \frac{1}{Z_4-z_1} & 0 \\ \frac{1}{Z_5-z_1} & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = 0, \quad \|\mathbf{w}\|_2 = 1. \quad (4.12)$$

The solution  $\mathbf{w}^T = [0, 1]$  solves the linear system exactly. If we use this solution in the non-linear equation (4.1) we get

$$B^{(2)}(z) = \frac{\sum_{n=1}^2 \frac{w_n f(z_n)}{z-z_n}}{\sum_{n=1}^2 \frac{w_n}{z-z_n}} = \frac{\frac{(0)1}{z-z_1} + \frac{(0)0}{z-z_2}}{\frac{(0)}{z-z_1} + \frac{(0)}{z-z_2}} = 0. \quad (4.13)$$

The critical interpolation at  $z_1 = Z_1$  has been lost leaving a useless constant function. This illustrates the importance of the *non-zero weights* requirement in equation 2.6. The loss of interpolation at  $z_1$  happens not only in this iteration but any subsequent iteration. In fact, if we continue the iteration, the third support point chosen is  $z_3 = Z_3$ , and the linearized least squares problem becomes

$$L\mathbf{w} = \begin{bmatrix} \frac{1}{Z_4-z_1} & 0 & 0 \\ \frac{1}{Z_5-z_1} & 0 & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = 0, \quad \|\mathbf{w}\|_2 = 1. \quad (4.14)$$

This problem has multiple exact solutions, for example  $\mathbf{w}^T = [0, 1, 0]$ ,  $\mathbf{w}^T = [0, 0, 1]$ , and  $\mathbf{w}^T = [0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]$ . All of these solutions result in the barycentric function  $B^{(3)}(z) = 0$ . Increasing the number of samples and or, changing the location of the samples, will not result in a better approximation.

**Remark.** *This example is dependent on the linearization used. When solving this problem in RKFIT, which uses a different linearization, with ten linearly chosen*

samples in  $[0, 1]$  and three initial poles at infinity, an approximation with one pole  $p = 3.09 \times 10^{-5}$  is found with  $\mathcal{H}_2$  error, as defined by equation (2.15), of  $2.7 \times 10^{-16}$ .

**Remark.** We note that in the AAA routine in Chebfun,  $B^{(2)}$  is returned as the final approximation with a reported error = 0. This happens because there is no check for zero weights before the error is calculated, and the error is only calculated at sample points  $Z^{(i)}$  not chosen as support points. This assumes that the interpolation property holds at all support points, but interpolation only happens at a support point  $z_n$  if the associated weight  $w_n$  is non-zero. Zero weights and their respective support points are removed after the AAA routine finds an approximation with sufficiently small error, in this case  $B^{(2)}$ , so the approximation returned is  $B(z) = 0$  having one support point  $z_2$  with  $w_2 = 1$  and  $f(z_2) = 0$ .

**Remark.** This example is very different from the failure of VF described in [38] where the number of sample points and their locations are critical to creating the fixed-point such that the VF iteration fails to converge. Adding a sixth sample point to the five points provided in the paper results in VF converging to a satisfactory solution. For sufficiently sampled inputs it seems unlikely that this fixed-point problem would occur.

#### 4.1.2 Example: miAAA on Six Functions

To illustrate the miAAA algorithm we use the samples from the six functions introduced in Chapter 1. The errors  $|f_k - B_k|$  of the miAAA approximations are shown in Figure 4.1. The miAAA algorithm for tolerance  $10^{-6}$  converged with 37 support points resulting in 36 poles.

#### 4.1.3 FastAAA

The FastAAA algorithm [23] was published in 2017 and contains very similar methodology to our MIMO extension in Section 4.1. A distinction between smiAAA

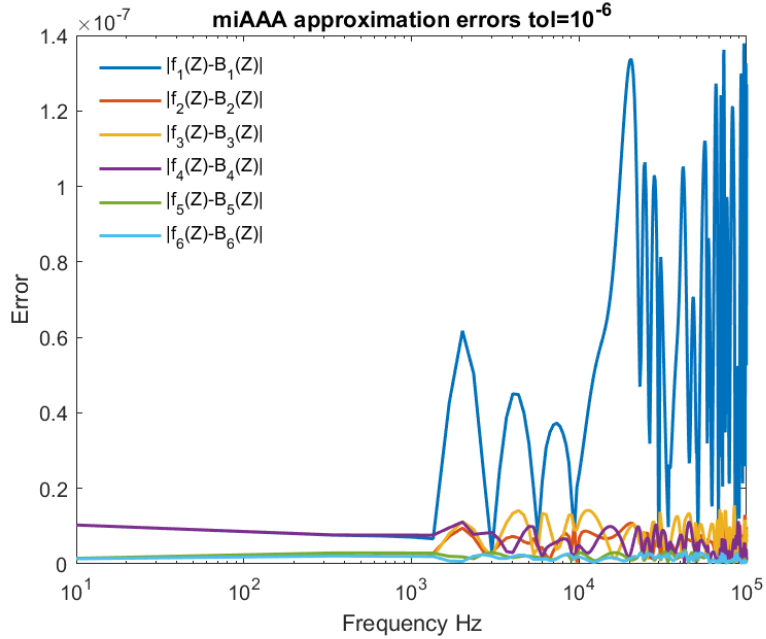


Figure 4.1: Errors for miAAA approximation of the six entries with input tolerance of  $10^{-6}$

and FastAAA is in the choice of support points. In FastAAA, the next support point  $z_i$  at each iteration is chosen as the sample point with the largest deviation from the current approximation for any of the input functions.

$$z_i = \arg \max_{z \in \mathbf{Z}^{(i)}} \left( \max_k |f_k(z) - B_k(z)| \right). \quad (4.15)$$

This methodology produces very similar results to miAAA, sometimes resulting in slightly fewer, and sometimes slightly more support points being chosen before convergence.

The FastAAA algorithm also provides a different methodology for building rational approximations with the additional constraint that all of the poles be stable. After the FastAAA algorithm converges to a barycentric approximation with error below the desired tolerance, the poles are computed and unstable poles are discarded.

A least squares problem is then solved to recover the best residues over the remaining stable poles. If the resulting stable partial fraction representation has deviation from the input above the desired threshold, then an additional support point is added to the barycentric representation, the stable poles and residues are recomputed and the deviation of the stable partial fraction is tested. This iteration repeats until a stable partial fraction representation with deviation below the desired tolerance is found. This differs from the smiAAA methodology described in chapter 5 where we compute the poles at every iteration and replace unstable poles with stable poles. In Section 5.1.1, we show an example where FastAAA fails to converge to a stable solution while the smiAAA methodology converges.

The FastAAA algorithm uses the same methodology as described in Section 4.1 to extend AAA to multiple functions. However computation of the SVD of the Löwner matrix in Equation (4.7) at each iteration is replaced with computing the SVD of a smaller matrix that is updated at each iteration. The updating scheme uses a Cholesky factor to update the singular values of the Löwner matrix without computing the SVD. This reduces the complexity of building the barycentric representation from  $\mathcal{O}(N^3)$  to  $\mathcal{O}(N^2)$  resulting in a significant computational improvement in run time.

#### 4.1.4 Hermitian Symmetry in LTI Applications

For model order reduction of LTI, as described in 2.5.1, where a Laplace transform has been applied to the underlying variable, as in equation 2.17, we will need to build approximations with Hermitian symmetry. That is, the poles and residues must come in complex conjugate pairs. We will use the following Hermitian symmetry enforcement from FastAAA on problems of this form. When a support point  $z_n$  is chosen we will add its complex conjugate  $z_{n+1} = \bar{z}_n$  as a support point as well. To ensure the residues come in complex conjugate pairs we enforce  $w_{n+1} = \bar{w}_n$ .

To achieve this, we introduce the complex to real operator

$$H(A) = \begin{bmatrix} \Re(A) & \Im(A) \\ \Re(A) & -\Im(A) \end{bmatrix}, \quad (4.16)$$

and compute  $L_H$ , the real analogue to  $L$  in equation (4.7),

$$L_H = \begin{bmatrix} H(l_1) & H(l_2) & \dots & H(l_3) \end{bmatrix}, \quad (4.17)$$

where  $l_1, l_2, \dots$  are the columns of  $L$ . The solution to the least squares problem in (4.7) can be found by solving the real-valued least squares problem

$$L_H \mathbf{w}_H = 0, \quad \text{s.t.} \quad \|\mathbf{w}_H\|_2 = 1, \quad (4.18)$$

and setting  $w_n = w_{H,n} + iw_{H,n+1}$  for  $n = 1, 3, 5, \dots, 2N - 1$ . To enforce the  $w_{n+1} = \bar{w}_n$  symmetry we restrict  $\mathbf{w}_H$  to the subspace of conjugate symmetric weight vectors

$$\mathbf{w}_H = S \tilde{\mathbf{w}}_H, \quad S = I_N \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & -1 \end{bmatrix}. \quad (4.19)$$

Where  $I_N$  denotes the  $N \times N$  identity matrix and  $\otimes$  denotes the Kronecker product. We can find  $\tilde{\mathbf{w}}_H$  from the SVD of  $L_H S$ , then recover  $\mathbf{w}_H = S \tilde{\mathbf{w}}_H$  and finally compute  $\mathbf{w}$  by  $w_n = w_{H,n} + iw_{H,n+1}$  for  $n = 1, 3, 5, \dots, 2N - 1$ .

#### 4.1.5 Set-Valued AAA

The set-valued AAA algorithm [28] for approximating NEPs as described in Section 2.5.2, published in 2018 also uses a very similar methodology to miAAA and FastAAA. It uses the same extension to multiple functions as FastAAA in Section 4.1.3 and as detailed in Section 4.1. It uses the same selection of support points as FastAAA, so there is only a slight difference between it and the miAAA methodology in Section 4.1. Set-valued AAA also uses an updating scheme instead of computing the SVD of the Löwner matrix in each iteration. This updating scheme is based on economy-size QR decompositions in contrast to the Cholesky decomposition in FastAAA. This results in set-valued AAA having a significantly faster running time than miAAA. However in [39] some numerical problems were reported using this updating scheme in place of the SVD. Details of these numerical issues were not discussed so we leave their consideration of the updating schemes in FastAAA and set-valued AAA to future research.

## CHAPTER FIVE

## METHODOLOGY FOR STABILITY ENFORCEMENT

5.1 Stability and Stable miAAA

In important applications of MIMO rational approximations on the imaginary axis, such as LTI described in Section 2.5.1, it is required that all the poles are stable. That is, the real part of each pole must be non-positive. In LTI, this requirement is a consequence of transforming the original time domain problem into a frequency domain problem, which is solved as a rational approximation problem on the imaginary axis. When transforming back into the original time domain, any pole with a positive real part results in an exponential function of positive exponent, causing unbounded growth. In EMT modeling, this would correspond to a non-physical system behavior where energy in the system increases exponentially instead of decaying over time.

Inspired by methodology found in both RKFIT and VF, we develop the functionality to (iteratively) enforce the stability of the poles in our approximations. To this end we prove the following theorem.

**Theorem 1.** *Let  $D(z) = \sum_n \frac{w_n}{z-z_n}$  be rational function with a simple zero at  $p \in \mathbb{C}$  ( $D(p) = 0$ ). Then the rational function  $D_p(z) = \sum_n \frac{w'_n}{z-z_n}$  with*

$$w'_n = w_n \frac{(z_n - q)}{(z_n - p)}. \quad (5.1)$$

*has a simple zero at  $q \in \mathbb{C}$  ( $D_p(q) = 0$ ) and  $D_p(p) \neq 0$ .*

*Proof:* Consider the barycentric representation

$$B(z) = \frac{J(z)}{D(z)} = \frac{\sum_n \frac{w_n f_n}{z - z_n}}{\sum_n \frac{w_n}{z - z_n}} \quad (5.2)$$

interpolating the values  $f_n$  at the support points  $z_n$ . We assume that  $B$  has an unstable, simple pole  $p$ . Thus,  $p$ , has positive real part, and is a simple zero of the rational function  $D$ . We show how to replace this unstable pole with a new stable pole  $q$ . We start by computing a new barycentric denominator  $D_p$  defined as

$$\begin{aligned} D_p(z) &= \frac{z - q}{z - p} D(z) = \left( \frac{z - p + p - q}{z - p} \right) \sum \frac{w_n}{z - p} \\ &= \left( 1 + \frac{p - q}{z - p} \right) \sum \frac{w_n}{z - z_n}. \end{aligned} \quad (5.3)$$

First, in order to deal with the  $z$  dependence outside of the summation, we compute

$$\begin{aligned} \frac{1}{z - p} \sum_n \frac{w_n}{z - z_n} &= \sum_n \frac{w_n}{(z - p)(z - z_n)} \\ &= \sum_n \frac{w_n}{z_n - p} \left( \frac{1}{z - z_n} - \frac{1}{z - p} \right) \\ &= \sum_n \left( \frac{w_n}{z_n - p} \right) \frac{1}{z - z_n} - \frac{1}{z - p} \sum_n \frac{w_n}{z_n - p} \\ &= \sum_n \left( \frac{w_n}{z_n - p} \right) \frac{1}{z - z_n}, \end{aligned} \quad (5.4)$$

as the second term in equation (5.4) is  $D(p) = 0$ . Plugging this last identity back

into equation (5.3) we obtain

$$\begin{aligned} \left(1 + \frac{p-q}{z-p}\right) \sum_n \frac{w_n}{z-z_n} &= \sum_n \frac{w_n}{z-z_n} + (p-q) \sum_n \left(\frac{w_n}{z_n-p}\right) \frac{1}{z-z_n} \\ &= \sum_n \left(1 + \frac{p-q}{z_n-p}\right) \frac{w_n}{z-z_n} \\ &= \sum_n \left(\frac{z_n-q}{z_n-p}\right) \frac{w_n}{z-z_n}. \end{aligned}$$

So  $D_p(z) = \sum_n \frac{w_{p,n}}{z-z_n}$ , where

$$w_{p,n} = \frac{z_n - q}{z_n - p} w_n, \quad (5.5)$$

is a proper rational function with the same poles,  $\{z_n\}$ , as  $D(z)$  and with residues  $w_{p,n}$  which are easily computed from the residues of  $D$ .

By also replacing the residues in the numerator of (5.2) with  $w_{p,n}$ , we define the stable barycentric approximation

$$\frac{\sum_n \frac{w_{p,n} f_n}{z-z_n}}{\sum_n \frac{w_{p,n}}{z-z_n}}$$

which, by equation (2.7) also interpolates the values  $f_n$  at the support points  $z_n$ .  $\square$

The stable miAAA algorithm (smiAAA) presented in this section, builds barycentric approximations in the same manner as the miAAA algorithm, but incorporates two additional steps. First, at each iteration, after the weight vector  $\mathbf{w}$  has been found by solving equation (4.7), the poles of the barycentric approximation are computed as detailed in Section 7.1. Second, each unstable pole  $p$  is replaced by it's stable reflection  $q = -1\Re(p) + \Im(p)$ , by computing a new weight vector  $\mathbf{w}'$  such that

$$w'_n = w_n \frac{(z_n - q)}{(z_n - p)}. \quad (5.6)$$

As described below, we experimented with different choices of  $q$ , not just the standard

choice of picking  $q$  as the reflection of  $p$  about the imaginary axis. We then compute the deviation and continue adding support points until the approximation attains the target error. Pseudocode of smiAAA is provided in Algorithm 5.4.

---

Algorithm 5.4: smiAAA

---

- 1: Set barycentric interpolants  $B_k = \text{mean}(f_k)$
  - 2: **while** deviation  $\geq$  tolerance **do**
  - 3: Select the next support point  $z_i = \arg \max_{z \in \mathbf{Z}^{(i)}} (\sum_k |f_k(z) - B_k(z)|)$
  - 4: Compute the Barycentric interpolant with linear least squares  
 $L\mathbf{w} = 0, \|\mathbf{w}\|_2 = 1$
  - 5: Compute the poles of  $B_k$  via equation (7.4)
  - 6: **for** poles  $p$  such that  $\Re(p) > 0$  **do**
  - 7: replace each unstable  $p$  with a stable  $q$  according to (5.6)
  - 8: **end for**
  - 9: Compute the deviation  $\max_k \|B_k(z) - f_k(z)\|_\infty$
  - 10: **end while**
  - 11: **return** barycentric interpolant
- 

### 5.1.1 Stable miAAA Example

Using a tolerance of  $10^{-6}$ , the fitting errors  $|f_k(z) - B_k(z)|_{z \in \mathbf{Z}}$  of the six functions introduced in Chapter 1 are shown in Figure 5.1. Here, unstable poles were replaced by their reflections about the imaginary axis. Notice that one function appears to have been more difficult to represent with stable poles as its approximation error is much larger than the other five. The smiAAA algorithm converged with 38 poles, a slight increase from the 36 poles obtained by the miAAA algorithm in Chapter 4. We see in this example the expected trade off when solving a constrained problem:

a larger number of poles may be necessary to approximate over stable poles. In this example, the miAAA algorithm selects three unstable poles, one of which has a large real magnitude.

**Remark.** *Based on various trials, we notice that the unstable poles that appear in the miAAA and FastAAA approximations do not disappear as the accuracy of miAAA and FastAAA is increased. In fact, using a tolerance of  $10^{-13}$ , miAAA selects seven unstable poles, some of which have large real magnitude. In contrast, the stable FastAAA [23] algorithm does not converge because of the appearance and persistence of these unstable poles. This example illustrates the advantage of the iterative replacement of unstable poles with their stable reflections vs. simply discarding unstable poles after convergence.*

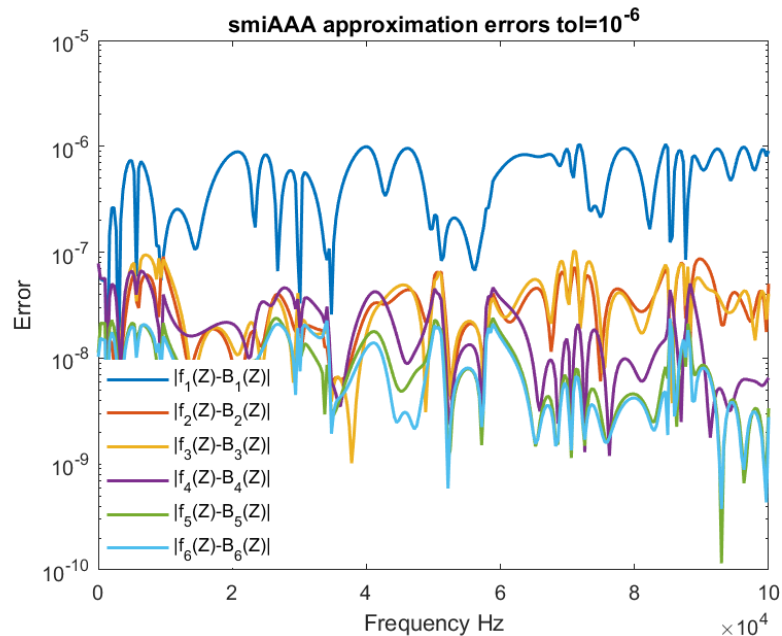


Figure 5.1: Logarithmic errors for smiAAA approximation of the six entries with input tolerance of  $10^{-6}$

### 5.1.2 Experimental Observations on the Reflection of the Poles

We experimented with replacing unstable poles  $p$  with stable choices other than simple reflection about the imaginary axis. Introducing a scaling parameter  $ref$  so that  $\Re(q) = -1 \cdot ref \cdot \Re(q)$ , the distance from the new pole to the imaginary axis can be changed. Figure 5.2 shows the convergence rate of smiAAA with tolerance  $10^{-8}$  on the six functions example from Chapter 1 using different values of  $ref$ . The convergence is fastest with  $ref = 1$  or  $ref = 2$ , and  $ref = 1$  has the smaller error at the final iteration. This is consistent with our experiments on other examples so we use  $ref = 1$  as the default for all calculations described in this document.

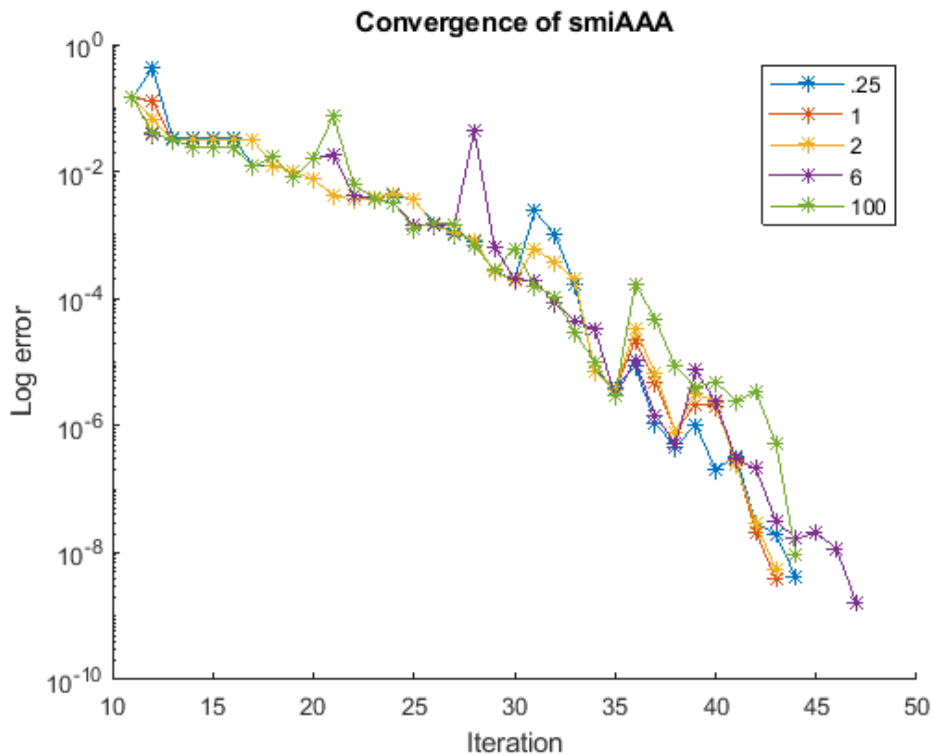


Figure 5.2: Logarithmic errors for smiAAA approximation at each iteration with different  $ref$  values plotted using different colors and tolerance  $10^{-8}$

### 5.1.3 Enforcing Other Pole Constraints

Although the pole relocation described in Section 5.1 was developed and tested primarily for enforcing the stability constraint on the poles, it is useful in other situations as well. Consider the nonlinear eigenvalue problem, as described in Section 2.5.2, in split form

$$F(z) = f(z)A, \quad z \in \Omega, \quad (5.7)$$

where  $f(z)$  is a holomorphic function on the complex domain  $\Omega$ . Depending on the selection of sample points, the set-valued AAA (as well as miAAA, weighted set-valued AAA, and FastAAA) may build non-holomorphic approximations of  $f(z)$ , that is approximations with poles in the interior of  $\Omega$ . Numerical issues may arise when computing an eigenvalue if an interior pole has been placed too close to it. To illustrate this issue, we consider a simple example and employ methodology similar to smiAAA to prevent poles from being placed in the interior of  $\Omega$ . At each iteration we compute the poles of the intermediate approximations, and if a pole  $p$  is found to be inside  $\Omega$ , we replace it with a pole  $q$  outside of the domain. The choice of  $q$  may depend on the geometry of the problem.

Consider the problem in equation (5.7) where  $\Omega$  is the unit disk and

$$f(z) = \left( \frac{14}{(z - 1.0001)(z - 1.00001)} + 1100z^3 \right), \quad A = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}. \quad (5.8)$$

We replace any interior pole  $p$  with an exterior pole  $q$  the reflection of  $p$  about the boundary, that is  $q = \frac{2 - \sqrt{|p|}}{\sqrt{|p|}}p$ . In Figure 5.3 we compare the poles obtained by set-valued AAA with those of smiAAA using this replacement of interior poles. Although both algorithms choose five poles, set-valued AAA obtained one pole in the interior

(left of the curve) of  $\Omega$ . We enlarge the area around the poles in Figure 5.4 for easier comparison.

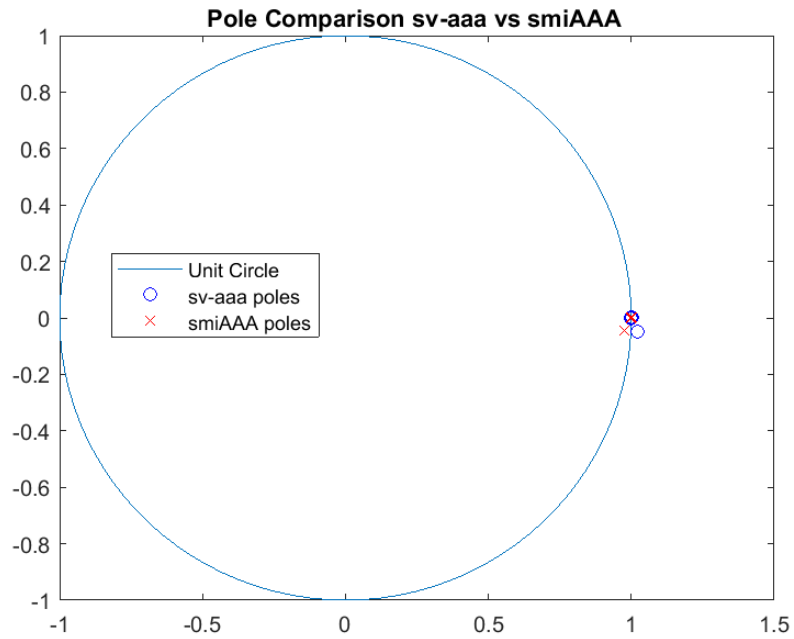


Figure 5.3: Location of poles chosen obtained by set-valued AAA (blue) vs. smiAAA (red) for a nonlinear eigenvalue problem

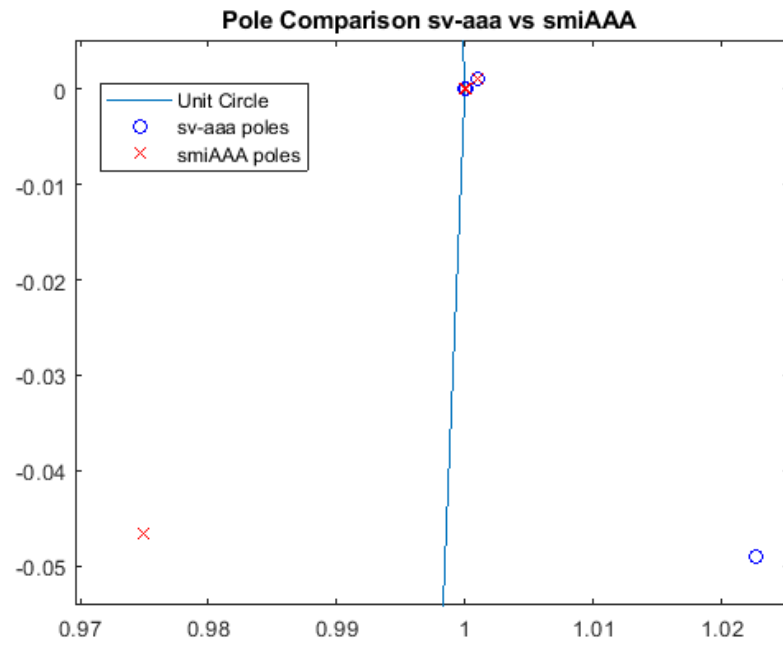


Figure 5.4: Location of poles (enlarged) obtained by set-valued AAA (blue) and smiAAA (red) for a nonlinear eigenvalue problem

## CHAPTER SIX

## LAWSON OPTIMIZATIONS

6.1 MIMO Lawson Optimization

After computing the miAAA approximation as described in Section 4.1, we extend the Lawson optimization described in [35] and Section 3.2.2 to the MIMO approximation in Section 4.1. We seek approximations such that all individual approximations are below the lowest possible common threshold. That is, we seek to minimize

$$\min \max_k \|f_k - B_k\|_\infty. \quad (6.1)$$

This is where the term *minimax approximation* gets its name. The Lawson optimization aims to make our approximations closer to *minimax*. We start by introducing non-interpolatory Barycentric representations for MIMO approximations

$$B_k^\wedge(z) = \frac{J_k^\wedge}{D^\vee} = \frac{\sum_{n=1}^N \frac{w_n^\wedge f_k(z_n)}{z - z_n}}{\sum_{n=1}^N \frac{w_n^\vee}{z - z_n}}. \quad (6.2)$$

Beginning with a set of Lawson weights  $\mathbf{l}\mathbf{w} = [1, 1, \dots, 1] \in \mathbb{R}^N$  all equal to one, we solve the weighted least squares problem

$$\min_{z \in Z} \left\| \sqrt{\mathbf{l}\mathbf{w}} [J_k^\wedge(z) - f_k(z) D^\vee(z)] \right\|_2, \quad \|\mathbf{w}^\wedge\|_2 + \|\mathbf{w}^\vee\|_2 = 1, \quad (6.3)$$

using the SVD of the weighted block non-interpolatory Löwner matrix

$$\text{diag}(\sqrt{\mathbf{l}\mathbf{w}^{\mathbf{K}}}) \begin{bmatrix} \text{diag}(f_1(\mathbf{Z}))C^l & -C^l\text{diag}(f_1(\mathbf{z})) \\ \text{diag}(f_2(\mathbf{Z}))C^l & -C^l\text{diag}(f_2(\mathbf{z})) \\ \vdots & \\ \text{diag}(f_K(\mathbf{Z}))C^l & -C^l\text{diag}(f_K(\mathbf{z})), \end{bmatrix} \quad (6.4)$$

where  $\mathbf{l}\mathbf{w}^{\mathbf{K}}$  indicates  $K$  copies of the Lawson weights (one for each function) and  $C^l$  is the modified Cauchy matrix described in equation 3.23. Each block

$$[\text{diag}(f_k(\mathbf{Z})C^l - C^l\text{diag}(f_k(\mathbf{z}))], \quad (6.5)$$

is the non-interpolatory Löwner matrix from equation (3.22) for function  $k$ . Additional importance is assigned to points where the approximation is worse by updating the Lawson weights

$$lw'_n = lw_n \cdot \max_k \|B_k^\wedge(Z_n) - f_k(Z_n)\|, \quad (6.6)$$

and then normalizing so that  $\|\mathbf{l}\mathbf{w}'\|_2 = 1$ . Returning to equation 6.3, iteration of the process continues until  $\|\mathbf{l}\mathbf{w}' - \mathbf{l}\mathbf{w}\|_\infty < tol$ . From experimentation we set  $tol = 10^{-8}$  as the default value. This iterative scheme does not produce monotone decreasing errors because of the linearization; as a consequence, the best approximation is often not from the final iteration. Pseudocode for the MIMO Lawson algorithm is given in Algorithm 6.5.

**Remark.** *We experimented with letting the Lawson weights change completely at each iteration to only capture information about the error at the current iteration; this was*

done by setting

$$lw_n = \max_k \|B_k^\wedge(Z_n) - f_k(Z_n)\|. \quad (6.7)$$

Although this did sometimes result in decreased errors, final results were worse than using equation (6.6) in all examples presented in this work. Additionally, we experimented with using the error from the interpolative barycentric representation to initialize the first Lawson weights. With the perfect interpolation at the support points, the Lawson weights corresponding to the support points would be initialized to zero (and therefore remain zero throughout the iterations). We perturbed these zero weights to small non-zero numbers allowing them to change during the iterations. Our testing has shown that initializing the Lawson weights with all ones produces equivalent or slightly better results.

**Remark.** *The fundamental idea of Lawson optimization, to increase the weight at sample points where the approximations are performing worse is similar to the AdaBoost methodology where new predictors are trained with greater weight on the samples where the current prediction is wrong.*

### 6.1.1 MIMO Lawson Example

We demonstrate the MIMO Lawson algorithm on the six function example introduced in Chapter 1 by optimizing the results from the miAAA approximation in Section 4.1.2. The max of the errors,  $\max_k |f_k(z) - B_k(z)|$ ,  $k = 1, 2, \dots, 6$ , with initial miAAA tolerance of  $10^{-6}$ , before and after ten iterations of the MIMO Lawson optimization, are shown in Figure 6.1. We indicate the result of using Lawson optimization by adding a 'L' to the algorithm name; thus, miAAA-L denotes the results of the miAAA algorithm after Lawson optimization. The resulting error  $\max_k |f_k(z) - B_k(z)|$  of miAAA-L is lower than that of miAAA and also appears

---

Algorithm 6.5: MIMO Barycentric Lawson Optimization

---

- 1: Initialize Lawson weights  $\mathbf{l}\mathbf{w} = [1, 1, \dots, 1]$
  - 2: **for** Max iterations **do**
  - 3: Compute weighted approximation
$$\min_{z \in Z} \left\| \sqrt{\mathbf{l}\mathbf{w}} [J_k^\wedge(z) - f_k(z) D^\vee(z)] \right\|_2, \quad \|\mathbf{w}^\wedge\|_2 + \|\mathbf{w}^\vee\|_2 = \mathbf{1}$$
via the SVD of equation (6.4)
  - 4: Compute the error  $\max_k \|B_k^\wedge - f_k\|_\infty$ , compare with current best approximation
  - 5: Update Lawson weights  $lw'_n = lw_n \cdot |B_k^\wedge(Z_n) - f_k(Z_n)|$
  - 6: **if**  $\|\mathbf{l}\mathbf{w}' - \mathbf{l}\mathbf{w}\|_\infty < tol$  **return** best approximation
  - 7: **end for**
  - 8: **return** best approximation
- 

to be more uniform.

**Remark.** *Although we are unaware of any results extending the classic Remez equioscillation property for minimax approximation [36] to MIMO problems, we note that the error  $\max_k |f_k(z) - B_k(z)|$  from smiAAA-L is much closer to equioscillatory than that of miAAA. The SISO barycentric Lawson optimization is used in [14] to find approximations closer to equioscillation before using a rational Remez algorithm to find barycentric minimax approximants.*

## 6.2 Stable Lawson Optimization

Both Lawson algorithms described in Section 3.2.2 and the MIMO Lawson algorithm described in Section 6.1 allow for the denominator weights  $\mathbf{w}^\vee$  to change during the iteration, which results in the poles being changed during the iteration. The initial non-interpolatory barycentric representation can (and will) have different

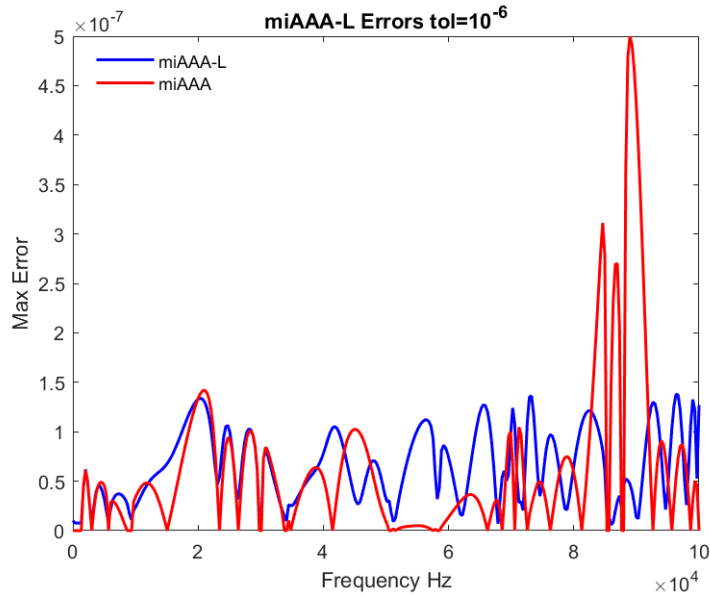


Figure 6.1: Max error  $\max_k |f_k(z) - B_k(z)|$ ,  $k = 1, 2, \dots, 6$ , for miAAA approximation of the six entries from the example in Section 4.1.2 with input tolerance of  $10^{-6}$  before and after ten iterations of the MIMO Lawson optimization

poles than the final interpolatory barycentric representation obtained from either AAA or miAAA. This changing of poles makes these algorithms poorly suited for applications where the stability of the poles is important such as LTI. We next describe a Lawson optimization algorithm where both of these problems are addressed to guarantee the stability of the poles in the final optimized representation.

Fixing the denominator weights  $\mathbf{w}^\vee = \mathbf{w}$  from the smiAAA algorithm in Section 5.1, the poles remain fixed and stable. We optimize only over the numerator weights  $\mathbf{w}^\wedge$  in the stable non-interpolatory Barycentric representation

$$B_k^\wedge(z) = \frac{J_k^\wedge}{D} = \frac{\sum_{n=1}^N \frac{w_n^\wedge f_k(z_n)}{z - z_n}}{\sum_{n=1}^N \frac{w_n}{z - z_n}}, \quad (6.8)$$

By solving the linear least squares problem

$$J_k^\wedge(z) = f_k(z)D(z). \quad (6.9)$$

This representation gives us the following linear system of equations for  $\mathbf{w}^\wedge$

$$\begin{bmatrix} \text{diag}(f_1(\mathbf{Z}))C^l \\ \text{diag}(f_2(\mathbf{Z}))C^l \\ \vdots \\ \text{diag}(f_K(\mathbf{Z}))C^l \end{bmatrix} \mathbf{w}^\wedge = \begin{bmatrix} C^l \text{diag}(f_1(\mathbf{z})) \\ C^l \text{diag}(f_2(\mathbf{z})) \\ \vdots \\ C^l \text{diag}(f_K(\mathbf{z})), \end{bmatrix} \mathbf{w}. \quad (6.10)$$

Unlike equation (6.4) this system is not a linearization of a non-linear system. We therefore do not need to constrain  $\mathbf{w}$  as  $\mathbf{w} = \mathbf{0}$  is not an extraneous solution to equation (6.10). Beginning with Lawson weights  $\mathbf{l}\mathbf{w} \in \mathbb{R}^N$  all initialized to one, we solve the weighted linear system

$$\text{diag}(\sqrt{\mathbf{l}\mathbf{w}}) \begin{bmatrix} C^l \text{diag}(f_1(\mathbf{z})) \\ C^l \text{diag}(f_2(\mathbf{z})) \\ \vdots \\ C^l \text{diag}(f_K(\mathbf{z})) \end{bmatrix} \mathbf{w}^\wedge = \text{diag}(\sqrt{\mathbf{l}\mathbf{w}}) \begin{bmatrix} \text{diag}(f_1(\mathbf{Z}))C^l \\ \text{diag}(f_2(\mathbf{Z}))C^l \\ \vdots \\ \text{diag}(f_K(\mathbf{Z}))C^l \end{bmatrix} \mathbf{w} \quad (6.11)$$

for  $\mathbf{w}^\wedge$  and  $\mathbf{l}\mathbf{w}$  is updated by

$$lw_n = lw_n \cdot \max_k |B_k^\wedge(Z_n) - f_k(Z_n)|. \quad (6.12)$$

The addition of the stability constraint reduces the number of parameters over which we can optimize, therefore we expect stable MIMO Lawson to provide less optimization than MIMO Lawson. Pseudocode for the stable MIMO Lawson

optimization is given in Algorithm 6.6.

---

Algorithm 6.6: Stable MIMO Barycentric Lawson Optimization

---

- 1: Initialize Lawson weights  $\mathbf{lw} = [1, 1, \dots, 1]$
  - 2: **for** Max iterations **do**
  - 3: Compute weighted approximation  $\sqrt{lw}[J_k^\wedge(z) = f_k(z)D(z)]$ ,
  - 4: Compute deviation  $\max_k \|B^\wedge - f_k\|_\infty$ , compare with current best approximation
  - 5: Update Lawson weights  $lw'_n = lw_n \cdot |B_k^\wedge(z_n) - f_k(z_n)|$
  - 6: **if**  $\|\mathbf{lw}' - \mathbf{lw}\|_\infty < 10^{-8}$  **return** best approximation
  - 7: **end for**
  - 8: **return** best approximation
- 

### 6.2.1 Stable MIMO Lawson Example

To demonstrate the stable MIMO Lawson Algorithm, we optimize the smiAAA results from Section 5.1.1 on the example introduced in Chapter 1. The max error,  $\max_k |f_k(z) - B_k(z)|$ ,  $k = 1, 2, \dots, 6$  with initial smiAAA tolerance of  $10^{-6}$  before and after ten iterations of the stable MIMO Lawson optimization are shown in Figure 6.2. Similar to the example of MIMO Lawson in Section 6.1.1 the  $\max_k |f_k(z) - B_k(z)|$  error has been decreased and is more uniform, as desired.

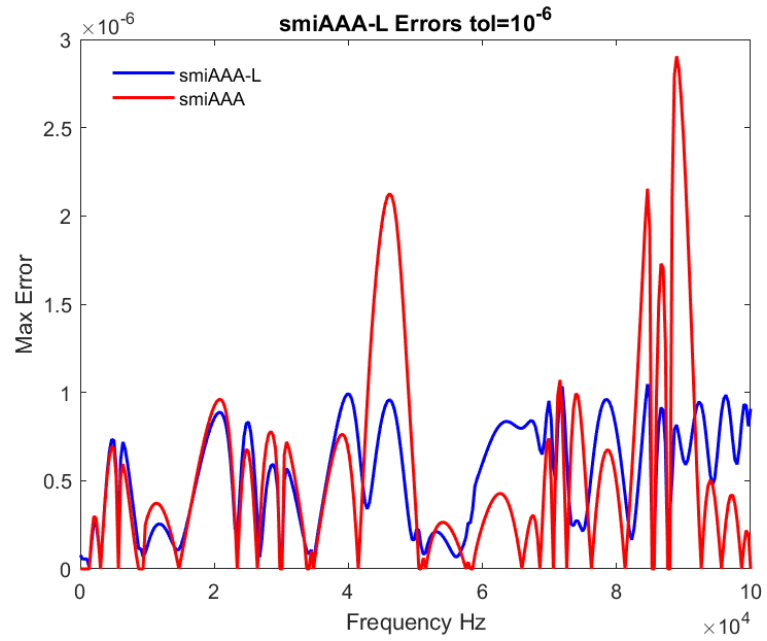


Figure 6.2: Max error  $\max_k |f_k(z) - B_k(z)|$ ,  $k = 1, 2, \dots, 6$  for smiAAA approximation of the six entries with input tolerance of  $10^{-6}$  before and after ten iterations of the stable MIMO Lawson optimization

## CHAPTER SEVEN

## CONVERSION TO PARTIAL FRACTIONS

Approximations in partial fraction form are important in applications including LTI and EMT modeling. By an inverse Laplace or Fourier transform, the partial fraction can be easily converted into a sum of exponential functions, which is exploited for fast computation by algorithms such as recursive convolution [20]. For convergence reasons, polynomial terms are converted into derivatives. In the MIMO case, the partial fractions may have different residues and polynomial coefficients for the approximation of each function  $f_k$ , but they will share a common set of poles. We now discuss converting the barycentric representation in equation (2.6) to a partial fraction representation

$$pf(z) = \sum_n \frac{\varphi_n}{z - p_n} + \sum \alpha_n z^n. \quad (7.1)$$

We do this by first computing the poles  $\{p_n\}$  as the zeros of the barycentric denominator. This is a non-linear root finding problem and is the most challenging step in recovering the partial fraction representation. The residues  $\{\varphi_n\}$  are then easily obtained by linear methods and, finally, a polynomial curve fitting is used to find the coefficients  $\{\alpha_n\}$ .

### 7.1 Computation of the Poles

The poles of the barycentric representation in (2.6) are the zeros of the barycentric denominator,

$$D(z) = \sum_{n=1}^N \frac{w_n}{z - z_n}, \quad (7.2)$$

where we assume that  $\{w_n\}$  and  $\{z_n\}$  are complex-valued, all  $\{z_n\}$  are distinct, and all  $\{w_n\}$  are non-zero. Equation (7.2) has a representation as the quotient of two polynomials,

$$\sum_{n=1}^N \frac{w_n}{z - z_n} = \frac{P(z)}{Q(z)}, \quad (7.3)$$

where  $Q(z) = \prod_n (z - z_n)$  and  $P \in \mathbb{P}^{N-1}$  has degree at most  $N-1$ . The desired zeros of equation (7.2) are therefore the roots of  $P$ . However, if the denominator  $Q(z)$  is very large at some  $z$  then equation (7.3) can be close to zero without  $P(z)$  being close to zero. To this end we compute the poles in a manner independent of the polynomial  $Q$ . Polynomial root finding problems are notorious for being ill-conditioned and sensitive to small perturbations in the parameters. Eigenvalue methods have proven to be an effective way to avoid ill-conditioning, so we formulate the problem  $P(z) = 0$  as an eigenvalue problem.

### 7.1.1 Case I: When $\deg(P)=N-1$

Beginning with the case where  $P$  is of degree  $N-1$ , we compute the zeros of equation (7.2) as the eigenvalues of the associated generalized eigenvalue problem, as implemented in the *prz* function in *Chebfun*,

$$Ax = \lambda Bx. \quad (7.4)$$

Here,  $A$  and  $B$  are given by

$$A = \begin{bmatrix} 0 & w_1 & w_2 & \dots & w_N \\ 1 & z_1 & & & \\ 1 & & z_2 & & \\ \vdots & & & \ddots & \\ 1 & & & & z_N \end{bmatrix}, \quad B = \text{diag}([0, 1, 1, \dots, 1]). \quad (7.5)$$

This formulation is called an *Arrowhead pencil matrix*, and it is closely related to the the diagonal plus rank one eigenvalue problem used in VF [24]. For details on the Arrowhead formulation for finding roots of rational functions see [25].

### 7.1.2 Case II: When $\deg(P) < N - 1$

We now consider the case where the degree of  $P < N - 1$ . If we rewrite equation (7.3) as a sum with common denominator  $Q$ ,

$$\frac{P}{Q} = \sum_{n=1}^N \frac{w_n}{z - z_n} \quad (7.6)$$

$$= \sum_{n=1}^N \frac{w_n \prod_{j=1}^N (z - z_j)/(z - z_n)}{\prod_{j=1}^N (z - z_j)} \quad (7.7)$$

$$= \frac{\sum_{n=1}^N w_n \prod_{j=1}^N (z - z_j)/(z - z_n)}{Q}, \quad (7.8)$$

we can identify the polynomial

$$P = \sum_{n=1}^N w_n \prod_{j=1}^N (z - z_j)/(z - z_n), \quad (7.9)$$

with leading order coefficient  $(\sum_n w_n)z^{N-1}$ . If  $\sum_n w_n = 0$ , then the leading order coefficient vanishes and  $\deg(P) \leq N - 2$ . In both the AAA and miAAA algorithms,

this can happen when the numerator in equation (2.6) is representing a polynomial of higher degree than that of the denominator. Numerically,  $\sum_n w_n$  will not be exactly zero in these cases, due to round off errors, but will instead be very small,  $\sum_n w_n = \varepsilon$ . This can result in a singular perturbation

$$\varepsilon z^{N-1} + P \in \mathbb{P}^{N-1}, \quad (7.10)$$

on  $P \in \mathbb{P}^{N-2}$  resulting in a spurious singular root and a perturbation on the poles. For a rigorous treatment of singular perturbation see, [29] (ch 2.2). To detect a singular perturbation before computing the poles, we first check if  $\sum_n w_n < tol$  for some small  $tol$ . For experimentation we set  $tol = 10^{-14}$  by default.

### 7.1.3 Deflating the Spurious Eigenvalues

If a singular perturbation is detected, instead of computing the zeros of  $D(z)$  we will compute the zeros of the rational function  $(z - z_j)D(z)$ , where  $z_j$  is one of the poles. We solve

$$0 = (z - z_j) D(z) = \sum_{n=1}^N \frac{w_n(z - z_j)}{z - z_n} \quad (7.11)$$

$$= \sum_{n=1}^N w_n - \sum_{n \neq j} \frac{w_n(z_n - z_j)}{z - z_n} \quad (7.12)$$

$$= - \sum_{n \neq j} \frac{w_n(z_n - z_j)}{z - z_n}, \quad (7.13)$$

and observe that  $(z - z_j)D(z)$  has the same zeros as  $D(z)$  since  $z_j$  cannot be a zero of  $D(z)$ . In fact  $D(z) \rightarrow \infty$  as  $z \rightarrow z_j$ . If  $\deg(P) = N - 2$ , then the representation in equation (7.13) no longer has a singular perturbation. If  $\deg(P) < N - 2$ , then  $\sum_{n \neq j} w_n$  must still be small, and we can repeat the process in equation (7.13) with another support point  $z'_j \in \{z_n\}_{n \neq j}$ , as many times as needed until  $\sum w_n > tol$ . The

rational function is now in the desired form and the eigenvalue problem can be solved to determine the poles. We refer to this routine as "prz deflated" (*przd*). We note that some of these spurious eigenvalues may be returned as `inf` by MATLAB; in this case they will be removed by the *prz* routine along with the two `inf` eigenvalues expected from the arrowhead matrix. In section 7.3.2, we detail an example where multiple deflations are performed, preventing the computation of spurious poles. The deflation of each spurious eigenvalue takes  $O(N)$  operations and computation of the eigenvalues by equation (7.4) is order  $O(N^3)$ ; thus *przd* also results in faster running time.

**Remark.** *Currently we select  $z_j$  as the support point with the largest absolute value. We are unaware of any examples where this produces unsatisfactory results. However, other choices of the support point  $z_j$  also produce satisfactory results and we leave the selection of the optimal  $z_j$  for future research.*

## 7.2 Computation of the Residues

Once the poles have been computed with equation (7.1), the residues are easily computed with the formula for residues for the quotient of analytic functions, see [8] (Sec. 6.76). Since  $D(p_j) = 0$  for all poles  $p_j$

$$\begin{aligned}
 \varphi_j &= \lim_{z \rightarrow p_j} (z - p_j)B(z) \\
 &= \lim_{z \rightarrow p_j} (z - p_j) \frac{J(z)}{D(z)} = \frac{N(p_j)}{\lim_{z \rightarrow p_j} \frac{D(z)}{z - p_j}} \\
 &= \frac{N(p_j)}{\frac{d}{dz}D(p_j)}. \tag{7.14}
 \end{aligned}$$

More explicitly:

$$\varphi_j = \frac{\sum_{n=1}^N \frac{w_n f(z_n)}{p_j - z_n}}{\sum_{n=1}^N \frac{w_n}{-(p_j - z_n)^2}}. \quad (7.15)$$

Defining the vector  $\boldsymbol{\varphi} = [\varphi_1, \varphi_2, \dots, \varphi_q]$ , where  $q$  is the number of poles, the vector containing the residues is calculated using

$$\boldsymbol{\varphi} = \frac{C \mathbf{w} \mathbf{f}}{C' \mathbf{w}}. \quad (7.16)$$

Where  $C$  is the Cauchy matrix from equation (3.17) and  $C'(i, j) = -C(i, j)^2$ ,  $\mathbf{w}$  is a vector containing the weights  $\{w_n\}$ , and  $\mathbf{w} \mathbf{f} = [w_1 f(z_1), w_2 f(z_2), \dots, w_N f(z_N)]$  is a vector containing the weights  $\{w_n\}$  multiplied component wise with the function values  $\{f(z_n)\}$  and the division in equation (7.16) is also component wise.

**Remark.** *This formulation can be adapted to the non-interpolatory barycentric representation in equation (3.20) by*

$$\boldsymbol{\varphi} = \frac{C \mathbf{w} \wedge \mathbf{f}}{C' \mathbf{w}^\vee}. \quad (7.17)$$

These Cauchy type matrices can be very ill-conditioned. However, we did not encounter any examples where this ill-conditioning produces poor results for our calculations.

### 7.3 Computation of the Polynomial Coefficients

With the poles  $\{p_n\}$  and the residues  $\{\varphi_n\}$  known, the final unknowns in equation (7.1) are the polynomial coefficients  $\{\alpha_n\}$ . We compute these using the built-in *polyfit*

function in MATLAB on the residual  $f(z) - \sum_n \frac{\varphi_n}{z-p_n}$ ,  $z \in \mathbf{Z}$  so that

$$\sum_n \alpha_n z^n \approx f - \sum_n \frac{\varphi_n}{z-p_n}, z \in \mathbf{Z}. \quad (7.18)$$

We can either specify the degree ( $\sum_n \alpha_n z^n$ ), if the application requires a maximal or minimal polynomial degree, or we can iterate over increasing degree, until a desired tolerance  $|r - f|$  is reached. This polynomial fitting problem can be ill-conditioned, but again we encountered no examples where this produces poor results. In Section 7.3.1 we demonstrate the conversion of the barycentric approximations from Section 4.1.2 to partial fraction form.

### 7.3.1 Conversion to Partial Fraction Example

The error  $|B_k(z) - pf_k(z)|$  of the six approximations from Section 4.1.2 with smiAAA tolerance of  $10^{-6}$  in barycentric form vs partial fraction form with a polynomial part of degree one, are shown in Figure 7.1. No deflations are performed and the 38 support points selected by smiAAA result in 37 poles in the partial fraction representation. Notice that the error is orders of magnitude below the  $10^{-6}$  tolerance.

### 7.3.2 Effects of Singular Perturbation

The example in Section 7.3.1 did not have any perturbations as described in Section 7.1. We demonstrate the effects of the perturbations in this example where spurious poles are returned by *prz*. We construct a function where the poles are known and the order of the numerator is larger than the order of the denominator so as to create the perturbation. For example, define

$$\vartheta(z) = \frac{(1.23 + z)(1 + z)(2 + z)(5 + z)(8 + z)^3}{(-3 + 2z + 2z^2)(1 + z^2)}, \quad (7.19)$$

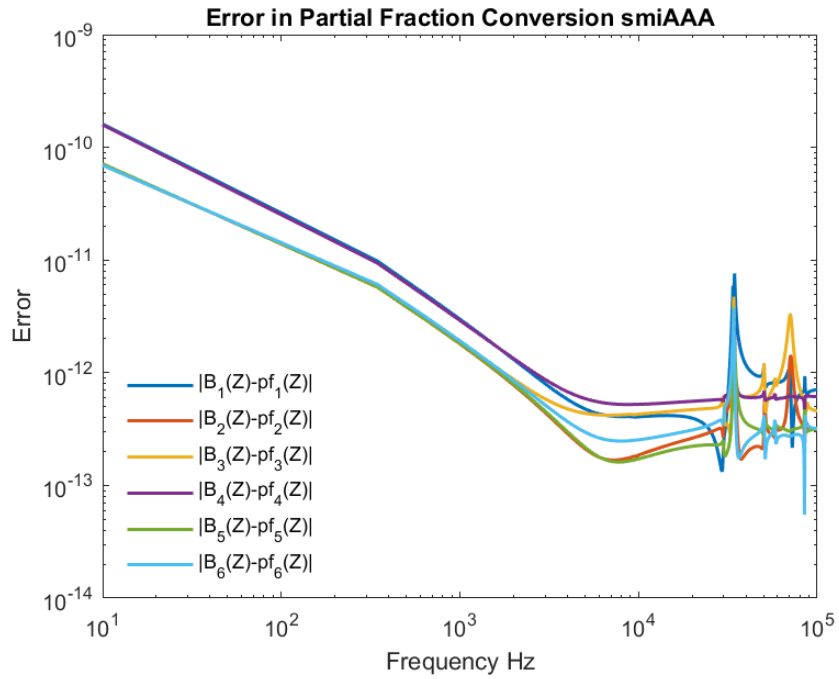


Figure 7.1: Logarithmic errors for smiAAA approximation of the six entries with input tolerance of  $10^{-6}$  in barycentric form vs partial fraction form

$1.0e-07^* -0.322739128 + 4.88291675i$
$1.0e-07^* 0.322741984 - 4.88291626i$
$-2.999999999998004 + 0.000000000000602i$
$0.999999999999999 + 0.000000000000003i$
$0.000000000000000 + 1.000000000000000i$
$-0.000000000000000 - 1.000000000000000i$

Table 7.1: Poles of  $\vartheta$  as computed with *prz*

$-3.000000000003228 + 0.000000000001496i$
$-0.000000000000511 - 0.99999999999858i$
$0.99999999994284 - 0.000000000005525i$
$0.000000000009701 + 1.000000000004371i$

Table 7.2: Poles of  $\vartheta$  as computed with *przd*

which is of type (7,4) and has poles  $\pm i, -3, 1$ . Approximating this function with AAA over 100 equispaced sample points in  $[-10, 10]$  achieves  $8.7e^{-12}$  error with 8 support points. The poles, computed with the *prz* algorithm are shown in Table 7.1. Two spurious poles are returned and a third spurious pole was removed by *prz* for being at infinity. The poles computed with *przd* are shown in Table 7.2, the algorithm performed three deflations. We see that all the spurious poles have been deflated by *przd*.

## CHAPTER EIGHT

## MODEL ORDER REDUCTION TEST PROBLEMS

In previous chapters we described all the methods necessary to consider the full model order reduction problem for LTI. We combine the smiAAA methodology in Section 5.1 with the enforcement of Hermitian symmetry from FastAAA, [23] discussed in Section 4.1.3, to build approximations with poles and residues appearing in complex conjugate pairs and all stable poles. We refer to this combination as symmetric smiAAA in the examples that follow. Because support points are chosen in pairs, the number of poles is usually odd, unless a deflation is performed, as outlined in Section 7.1. At least one pole will therefore have  $\Im(p) = 0$ , so it does not come in a pair. We can then convert the barycentric approximations to partial fraction representations as would be required to analytically invert the Laplace transform and obtain a representation via exponential functions (see Section 2.5.1).

For comparison, we first run smiAAA to the desired initial tolerance and then run 10 Lawson iterations (smiAAA-L). VF and RKFIT will be initialized with the same number of poles as selected by symmetric smiAAA. The VF algorithm was specifically designed for this kind of problem and is widely used in industry for this purpose. We use the *VFdriver* routine from the Matrix Fitting Toolbox for the VF results presented here.

Functionality for solving these problems is also implemented in RKFIT; specifically, we will use the real arithmetic option in RKFIT, as the authors recommend for these symmetric problems. RKFIT is initialized in two ways, once with the IEEE poles routine and once with all poles at  $\infty$ . The initialization of RKFIT with all poles at  $\infty$  gives RKFIT no information about where the poles should be located. This

is a reasonable comparison with smiAAA since smiAAA has no *a priori* information about the location of the poles. In [32], methods are discussed for estimating the number of initial poles required for VF to reach a desired accuracy. Running times are computed with the *timeit* routine in MATLAB on an AMD Ryzen PRO 2500U.

### 8.1 Example: Vector Fitting Users Guide

We now consider the full model order reduction problem for the power systems example introduced in Chapter 1. We have been using the first six of the full systems 36 functions. We now approximate all 36 functions with Hermitian symmetry and stable poles. The symmetric smiAAA algorithm with tolerance  $10^{-6}$  selects 67 poles. Table 8.1 gives the  $\mathcal{H}_2$ ,  $L_2$  and  $L_\infty$  errors as well as the running times for the three algorithms considered. Figure 8.1 shows a comparison of the approximations. In this example we see that smiAAA, and smiAAA-L yield better  $\mathcal{H}_2$  errors than VF and an order of magnitude better results in  $L_\infty$ , VF produces a slightly lower  $L_2$  error. RKFIT (IEEE poles) outperforms both smiAAA and VF and the very best results are produced by RKFIT ( $\infty$  poles). The Lawson optimization on this example resulted in both a reduction in  $L_\infty$  as well as  $L_2$  and  $\mathcal{H}_2$  error even though the  $L_2$  and  $\mathcal{H}_2$  errors are not considered by the algorithm. In Figure 8.2 we compare the number of poles chosen by FastAAA and symmetric smiAAA on this example. We see that FastAAA chooses slightly fewer parameters when requesting  $10^3$  and  $10^4$  final accuracy. As the final desired accuracy is increased symmetric smiAAA chooses fewer parameters than FastAAA and converges to accuracies where FastAAA does not converge.

Algorithm	$\mathcal{H}_2$ error	$L_2$ error	$L_\infty$ error	running time (sec)
symmetric smiAAA	$1.46e^{-6}$	$1.59e^{-7}$	$9.86e^{-7}$	2.70
symmetric smiAAA-L	$1.36e^{-6}$	$1.48e^{-7}$	$5.01e^{-7}$	5.30
VF	$1.68e^{-6}$	$1.41e^{-7}$	$1.42e^{-6}$	1.98
RKFIT(IEEE poles)	$1.15e^{-6}$	$1.26e^{-7}$	$1.00e^{-6}$	2.02
RKFIT( $\infty$ poles)	$5.75e^{-7}$	$6.28e^{-8}$	$4.88e^{-7}$	1.97

Table 8.1: Errors and running times for different algorithms on the Vector Fitting Users Guide example with 67 poles

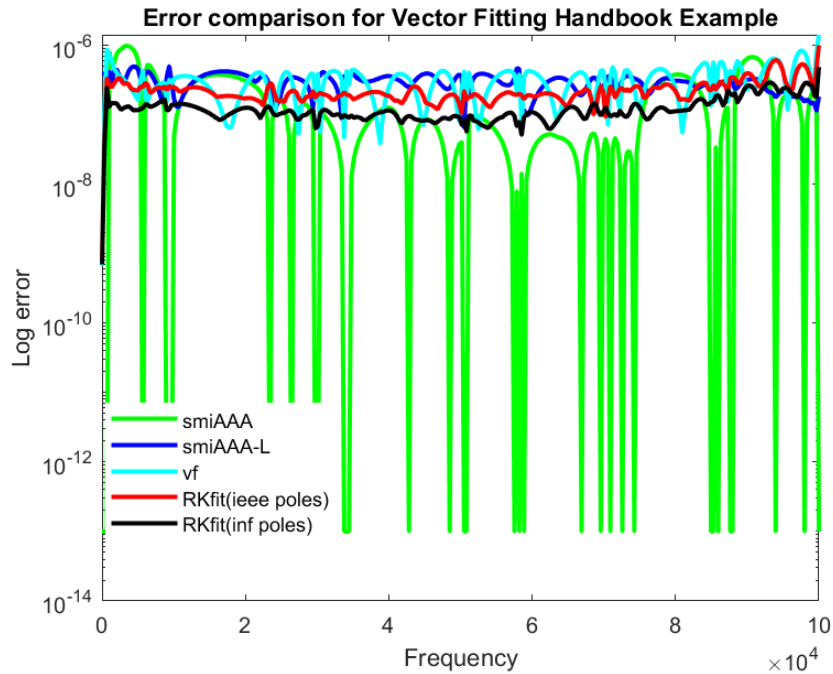


Figure 8.1: Errors for different algorithms on Vector Fitting Users Guide example

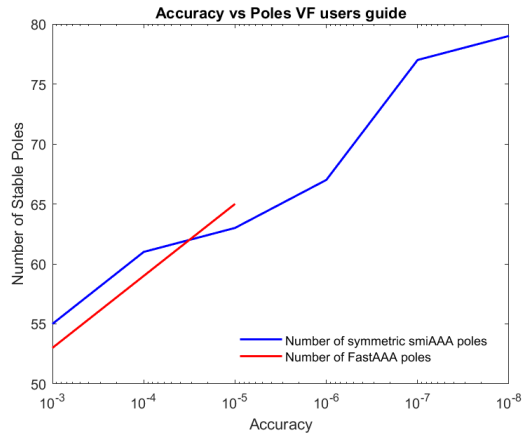


Figure 8.2: Number of poles chosen by symmetric smiAAA and FastAAA for different accuracies on VF users guide example

## 8.2 Example:Matrix Fitting Toolbox

The following example using nine functions can be found in the Matrix Fitting Toolbox package as ex2\_Y.m. This example comes from a power system problem (similar to Section (8.1)), for which the symmetric smiAAA algorithm requires 55 poles to achieve  $10^{-4}$  accuracy. The approximation errors are given in Figure 8.4 and a comparison of errors and running times are shown in Table 8.2. The Lawson optimization again resulted in decreased error, in the  $\mathcal{H}_2$ ,  $L_2$  and  $L_\infty$  errors. We see in this example that smiAAA-L produced the best results in all error metrics. RKFIT ( $\infty$  poles) also converged to a relatively comparable solution but RKFIT (IEEE poles) converged to a much less accurate solution. This highlights the advantage of being able to choose the target error in the smiAAA algorithm. In Figure 8.3 we see the number of poles chosen by FastAAA vs symmetric smiAAA. We see that symmetric smiAAA chooses the same number or fewer poles at every accuracy.

Algorithm	$\mathcal{H}_2$	$L_2$ error	$L_\infty$ error	running time (sec)
symmetric smiAAA	$3.91e^{-4}$	$2.80e^{-6}$	$1.60e^{-5}$	0.62
symmetric smiAAA-L	$3.31e^{-4}$	$2.37e^{-6}$	$5.00e^{-6}$	1.23
VF	$3.82e^{-4}$	$2.73e^{-6}$	$1.28e^{-5}$	0.80
RKFIT(IEEE poles)	$1.9e^{-3}$	$1.354e^{-5}$	$1.49e^{-4}$	1.08
RKFIT( $\infty$ poles)	$9.03e^{-4}$	$6.43e^{-6}$	$3.60e^{-5}$	2.49

Table 8.2: Errors and running times for different algorithms on MFT example with 55 poles

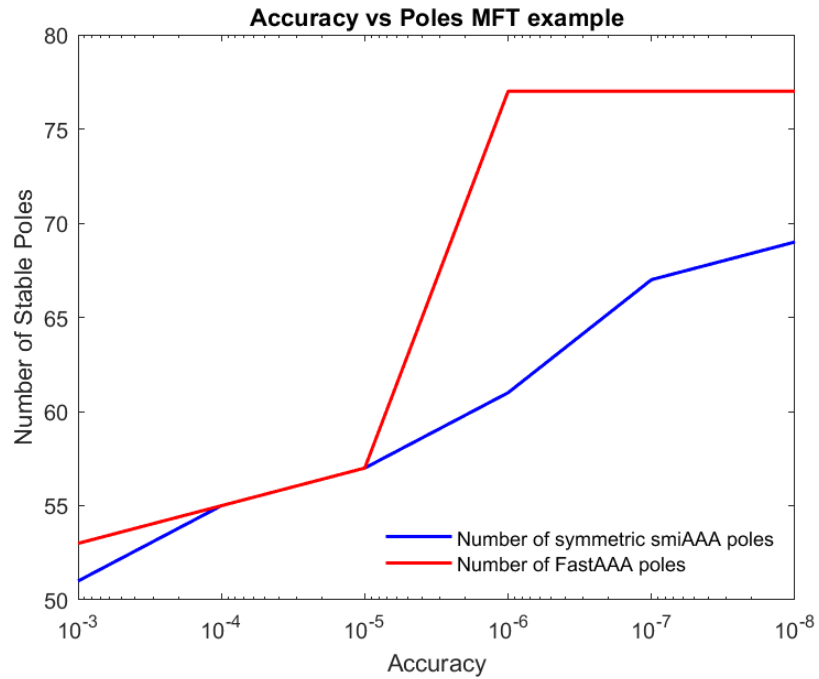


Figure 8.3: Errors for different algorithms on MFT example

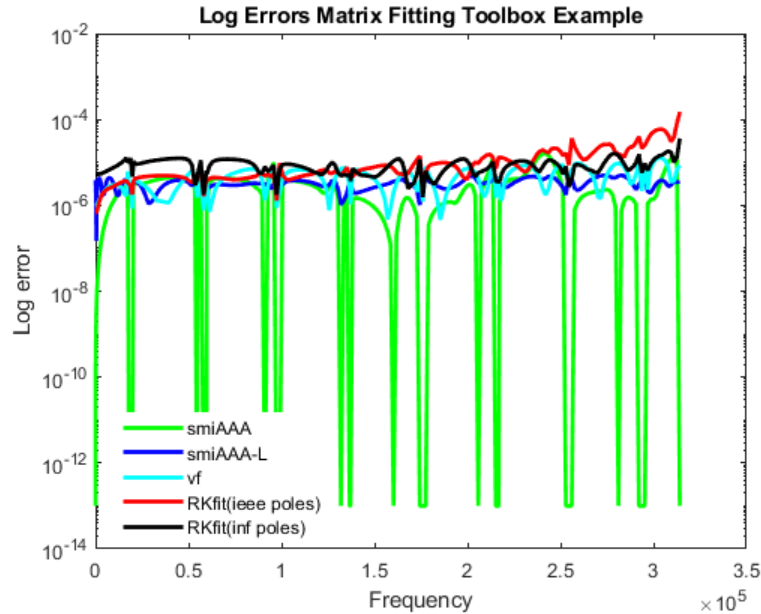


Figure 8.4: Number of poles chosen by symmetric smiAAA and FastAAA for different accuracies on the MTF example

### 8.3 Example: International Space Station

The following 9 function example comes from a vibration model for the international space station from [17]; the data is available from [10]. Our smiAAA algorithm chooses 49 poles. The logarithmic errors are given in Figure 8.5 and a comparison of errors and running times are shown in Table 8.3. In this example, the best results come from RKFIT ( $\infty$  poles), with smiAAA producing results of the same order magnitude while VF converged to a solution with  $\mathcal{H}_2$ ,  $L_2$ - and  $L_\infty$  errors an order of magnitude worse. In contrast to the previous examples, the Lawson optimization resulted in an increased  $\mathcal{H}_2$  and  $L_2$  errors. The  $\mathcal{H}_2$  error is the more important metric for LTI, so the Lawson optimization made the approximation worse for this example. In practice, the result of the Lawson optimization is compared with the one without optimization, before adopting the one with the smaller error.

Algorithm	$\mathcal{H}_2$	$L_2$ error	$L_{inf}$ error	running time (sec)
symmetric smiAAA	$2.5e^{-3}$	$4.84e^{-6}$	$9.03e^{-5}$	1.27
symmetric smiAAA-L	$5.3e^{-3}$	$1.03e^{-5}$	$8.14e^{-5}$	2.38
VF	$5.66e^{-2}$	$1.10e^{-4}$	$5.40e^{-3}$	1.16
RKFIT(IEEE poles)	$1.4e^{-3}$	$2.69e^{-6}$	$7.56e^{-5}$	1.25
RKFIT( $\infty$ poles)	$1.1e^{-3}$	$2.69e^{-6}$	$5.77e^{-5}$	1.23

Table 8.3: Errors and running times for different algorithms on the international Space Station example with 49 poles

In Figure 8.6, we compare the number of poles chosen by FastAAA and symmetric smiAAA on this example. FastAAA chooses slightly fewer poles than symmetric smiAAA at all accuracies except the final  $10^{-8}$  accuracy.

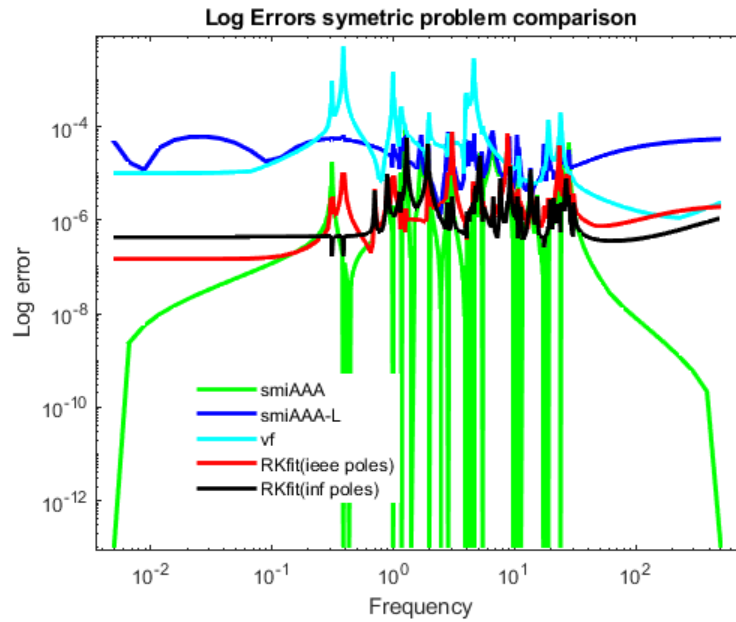


Figure 8.5: Errors for different algorithms on the International Space Station example

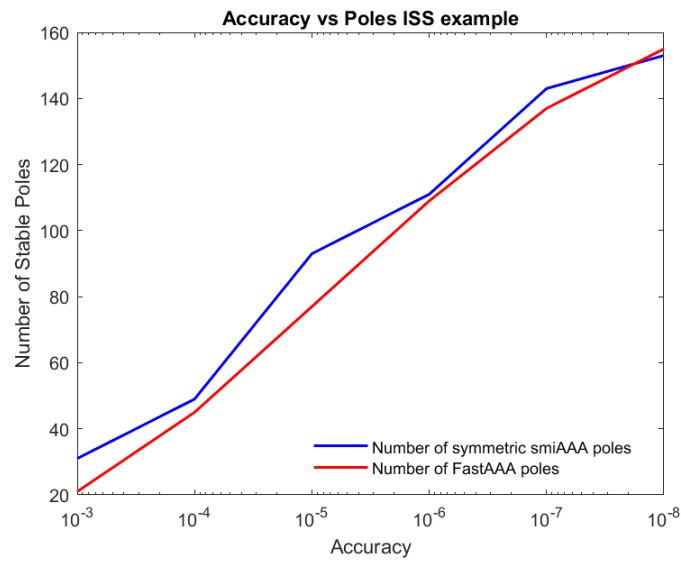


Figure 8.6: Errors for different algorithms on the ISS example

## CHAPTER NINE

## CONCLUSIONS AND FUTURE WORK

In this work, we have extended the functionality of the SISO AAA algorithm to handle the approximation of MIMO problems. The resulting miAAA algorithm proved to be fast, robust, and produced approximations of low order. This algorithm can be used in nonlinear eigenvalue problems to build linear systems whose eigenvalues approximate those of the nonlinear problem. This work is very similar to the set-valued AAA and weighted set-valued AAA algorithms, which were designed for these problems, often producing the same or very similar results.

We developed a methodology for enforcing the stability constraint on the shared poles of the MIMO approximations. This constraint is critical for the model order reduction of LTI. This methodology, which emulates that found in VF and RKFIT and included in the smiAAA algorithm, proved effective on all examples we tested. We demonstrated that the smiAAA algorithm converges to a stable solution on problems where the FastAAA algorithm did not converge, highlighting the strength of smiAAA. This methodology may be applicable in other applications, where the location of the final poles is important. Poles can be moved in or out of other domains, allowing for different constraints to be enforced.

To find MIMO approximations closer to minimax, we extended the SISO barycentric Lawson optimization to our MIMO approximations. This post-processing returns approximations with smaller and more uniform error than the initial miAAA approximations. For applications such as LTI, where the stability of the poles is required, we introduced a stable MIMO barycentric Lawson optimization, which returns approximations with smaller and more uniform error than the initial smiAAA

approximations. As far as we know, this is the only MIMO minimax optimization algorithm.

These new tools, combined with the enforcement of Hermitian symmetry introduced by FastAAA, allow for the model order reduction of LTI with the symmetric smiAAA algorithm. We demonstrated examples where symmetric smiAAA algorithm produced better results than the industry standard VF algorithm for these problems in EMT programs. The relatively new RKFIT algorithm outperformed both symmetric smiAAA and VF on these problems but symmetric smiAAA still offers an advantage, in that it requires no initial guess for the order of the approximation to guarantee a desired accuracy. It may also be possible to significantly reduce the running time of symmetric smiAAA. Symmetric smiAAA has two computationally expensive steps, the SVD of the Löwner matrix and the computation of the poles. By implementing an updating scheme similar to those in FastAAA or set-valued AAA, we could reduce the computation time of the SVD step. Fast methods have also been suggested for computing the eigenvalues of arrowhead matrices, so the pole computation could be sped up by implementing one of these methods. Reduction in the running time of smiAAA would make it more desirable over other methods. We leave the stability analysis and experimentation with these methods to future research.

## REFERENCES CITED

- [1] A. C. Antoulas. *Approximation of Large-Scale Dynamical Systems (Advances in Design and Control) (Advances in Design and Control)*. Society for Industrial and Applied Mathematics, USA, 2005.
- [2] A. C. Antoulas, D. C. Sorensen, and S. Gugercin. A survey of model reduction methods for large-scale systems. *Contemporary Mathematics*, 280:193–219, 2001.
- [3] M. Berljafa. *Rational Krylov Decompositions: Theory and Applications*. PhD thesis, 01 2017.
- [4] M. Berljafa and S. Güttel. Generalized rational Krylov decompositions with an application to rational approximation. *SIAM Journal on Matrix Analysis and Applications*, 36(2):894–916, 2015.
- [5] M. Berljafa and S. Güttel. The RKFIT algorithm for nonlinear rational approximation. *SIAM Journal on Scientific Computing*, 39(5):A2049–A2071, 2017.
- [6] A. S. Bernd Girod, Rudolf Rabenstein. *Signals and systems*. Wiley, 2<sup>nd</sup> edition, 2001.
- [7] D. Braess. *Nonlinear Approximation Theory*. Springer-Verlag, Berlin, 1986.
- [8] J. Brown and R. Churchill. *Complex Variables and Applications*. McGraw-Hill, 8<sup>th</sup> edition, 2009.
- [9] A. L. Cauchy. *Exercices d’analyse et de physique mathématique*. Bachelier, Paris, 1840.
- [10] Y. Chahlaoui and P. Van Dooren. Benchmark examples for model reduction of linear time-invariant dynamical systems. In P. Benner, D. C. Sorensen, and V. Mehrmann, editors, *Dimension Reduction of Large-Scale Systems*, pages 379–392, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
- [11] D. Deschrijver, B. Gustavsen, and T. Dhaene. Advancements in iterative methods for rational approximation in the frequency domain. *IEEE Transactions on Power Delivery*, 22(3):1633–1642, 2007.
- [12] D. Deschrijver, M. Mrozowski, T. Dhaene, and D. De Zutter. Macromodeling of multiport systems using a fast implementation of the vector fitting method. *IEEE Microwave and Wireless Components Letters*, 18(6):383–385, 2008.
- [13] S. Elsworth and S. Güttel. Conversions between barycentric, rkfun, and Newton representations of rational interpolants, 2017.

- [14] S. Filip, Y. Nakatsukasa, L. Trefethen, and B. Beckermann. Rational minimax approximation via adaptive barycentric representations. *SIAM Journal on Scientific Computing*, 40(4):A2427–A2455, 2018.
- [15] G. H. Golub and G. Meurant. *Matrices, Moments, and Quadrature*. 1994.
- [16] S. Gugercin, A. Antoulas, and C. Beattie. H2 model reduction for large-scale linear dynamical systems. *SIAM J. Matrix Anal. Appl.*, 30:609–638, 2008.
- [17] S. Gugercin, A. C. Antoulas, and N. Bedrossian. Approximation of the international space station 1r and 12a models. In *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228)*, volume 2, pages 1515–1516 vol.2, 2001.
- [18] B. Gustavsen. Improving the pole relocating properties of vector fitting. *IEEE Transactions on Power Delivery*, 21(3):1587–1592, 2006.
- [19] B. Gustavsen. User’s guide for vectfit3.m. *SINTEF Energy Research, Trondheim, Norway*, 2008.
- [20] B. Gustavsen and H. M. J. De Silva. Inclusion of rational models in an electromagnetic transients program: Y-parameters, z-parameters, s-parameters, transfer functions. *IEEE Transactions on Power Delivery*, 28(2):1164–1174, 2013.
- [21] B. Gustavsen and A. Semlyen. Rational approximation of frequency domain responses by vector fitting. *IEEE Transactions on Power Delivery*, 14(3):1052–1061, 1999.
- [22] S. Güttel and F. Tisseur. The nonlinear eigenvalue problem. *Acta Numerica*, 26:1–94, 2017.
- [23] A. Hochman. FastAAA: A fast rational-function fitter. In *2017 IEEE 26th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, pages 1–3, 2017.
- [24] N. Jakovčević Stor, I. Slapničar, and J. L. Barlow. Accurate eigenvalue decomposition of real symmetric arrowhead matrices and applications. *Linear Algebra and its Applications*, 464:62–89, Jan 2015.
- [25] P. Lawrence. Eigenvalue methods for interpolation bases. 2013.
- [26] S. Lefteriu and A. C. Antoulas. A new approach to modeling multiport systems from frequency-domain data. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(1):14–27, 2010.

- [27] S. Lefteriu and A. C. Antoulas. On the convergence of the vector-fitting algorithm. *IEEE Transactions on Microwave Theory and Techniques*, 61(4):1435–1443, 2013.
- [28] P. Lietaert, J. Pérez, B. Vandereycken, and K. Meerbergen. Automatic rational approximation and linearization of nonlinear eigenvalue problems. *eprint 1801.08622*, 2018.
- [29] J. D. Logan. *Applied Mathematics*. Wiley, 3<sup>rd</sup> edition, 2006.
- [30] K. Löwner. Über monotone Matrixfunktionen. 38:177–216, 1934.
- [31] L. Monzón, W. Johns, S. Iyengar, M. Reynolds, J. Maack, and K. Prabakar. A multi-function AAA algorithm applied to frequency dependent line modeling. In *2020 IEEE Power Energy Society General Meeting (PESGM)*, pages 1–5, 2020.
- [32] J. Morales, J. Mahseredjian, A. Ramirez, K. Sheshyekani, and I. Kocar. A Löwner/MPM—VF combined rational fitting approach. *IEEE Transactions on Power Delivery*, 35(2):802–808, 2020.
- [33] J. Morales-Rodriguez, E. Medina, J. Mahseredjian, A. Ramirez, K. Sheshyekani, and I. Kocar. Frequency-domain fitting techniques: A review. *IEEE Transactions on Power Delivery*, pages 1–1, 2019.
- [34] A. Mouhaidali, D. Tromeur DERVOUT, O. Chadebec, J. M. Guichon, and S. Silvant. Electromagnetic transient analysis of transmission line based on rational krylov approximation. *IEEE Transactions on Power Delivery*, pages 1–1, 2020.
- [35] Y. Nakatsukasa, O. Sète, and L. Trefethen. The AAA algorithm for rational approximation. *SIAM Journal on Scientific Computing*, 40(3):A1494–A1522, 2018.
- [36] E. Y. Remez. Sur la détermination des polynômes d’approximation de degré donné. 10(41), 1934.
- [37] J. Rice and K. Usow. The Lawson algorithm and extensions. 22(101):118–127, 1968.
- [38] G. Shi. On the nonconvergence of the vector fitting algorithm. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 63(8):718–722, 2016.
- [39] G. Stefan, N. Porzio, G. Maria, and T. Françoise. Robust rational approximations of nonlinear eigenvalue problems. *Preprint, The University of Manchester*, 2020.
- [40] L. N. Trefethen. *Approximation Theory and Approximation Practice, Extended Edition*. 2019.