On Computing Zeros of Analytic Functions and Related Problems in Structured Numerical Linear Algebra

Jury:
Prof. Dr. ir. E. Aernoudt, voorzitter
Prof. Dr. ir. A. Haegemans, promotor
Prof. Dr. ir. M. Van Barel, promotor
Prof. Dr. D. A. Bini
(Università degli Studi di Pisa)
Prof. Dr. ir. R. Cools
Prof. Dr. J. Quaegebuer
Prof. Dr. ir. S. Van Huffel
Prof. Dr. M. N. Vrahatis
(University of Patras)

Proefschrift voorgedragen tot het behalen van het doctoraat in de toegepaste wetenschappen
door

Peter KRAVANJA

U.D.C. 519.6, 681.3\*G13, 681.3\*G15, 517.53

maart 1999
Computers are useless, they can only give you answers.

—Pablo Picasso
On Computing Zeros of Analytic Functions and Related Problems in Structured Numerical Linear Algebra

Peter Kravanja

ABSTRACT. This thesis is a blend of computational complex analysis and numerical linear algebra. We study the problem of computing all the zeros of an analytic function that lie inside a Jordan curve. The algorithm that we present computes not only approximations for the zeros but also their respective multiplicities. It does not require initial approximations for the zeros and we have found that it gives accurate results. A Fortran 90 implementation is available (the package ZEAL). Our approach is based on numerical integration and the theory of formal orthogonal polynomials. We show how it can be used to locate clusters of zeros of analytic functions. In this context we also present an alternative approach, based on rational interpolation at roots of unity. Next we consider the related problem of computing all the zeros and poles of a meromorphic function that lie inside a Jordan curve and that of computing all the zeros of an analytic mapping (in other words, all the roots of a system of analytic equations) that lie in a polydisk. We also consider analytic functions whose zeros are known to be simple, in particular Bessel functions (the package ZEBEC) and certain combinations of Bessel functions. Next we propose a modification of Newton's method for computing multiple zeros of analytic mappings. Under mild assumptions our iteration converges quadratically. It involves certain constants whose product is a lower bound for the multiplicity of the zero. As these constants are usually not known in advance, we devise an iteration in which not only an approximation for the zero is refined, but also approximations for these constants. In the last part of this thesis we develop stabilized fast and superfast algorithms for rational interpolation at roots of unity. These algorithms lead to fast and superfast solvers for (indefinite) linear systems of equations that have Hankel or Toeplitz structure.
# Contents

Acknowledgements .................................................. ix

Nederlandse samenvatting ........................................ xı

Outline of the thesis ................................................. 1

## Part 1.

### Chapter 1. Zeros of analytic functions
1. Introduction .................................................. 5
2. Formal orthogonal polynomials ............................... 14
3. An accurate algorithm to compute zeros of FOPs ............ 23
4. Numerical examples .......................................... 28
5. The software package ZEAL ................................... 35
6. A derivative-free approach ................................... 49
Acknowledgements .................................................. 56

### Chapter 2. Clusters of zeros of analytic functions
1. How to obtain the centre of a cluster and its weight ......... 57
2. A numerical example .......................................... 60
3. Rational interpolation at roots of unity ....................... 64
4. More numerical examples ...................................... 72
Acknowledgements .................................................. 74

### Chapter 3. Zeros and poles of meromorphic functions
1. Introduction .................................................. 77
2. Theoretical considerations and numerical algorithm ........ 78
3. A numerical example .......................................... 81

### Chapter 4. Systems of analytic equations
1. Introduction .................................................. 83
2. A multidimensional logarithmic residue formula ............. 85
3. The algorithm ................................................ 88
4. Numerical examples .......................................... 92
Acknowledgements .................................................. 94

## Part 2.

### Chapter 5. Computing simple zeros of Bessel functions

vii
Complex analysis has intrigued me from the moment that I first heard about it. I can still see myself watching BBC Open University programs about line integrals in the complex plane, residues and Cauchy’s Theorem on early Sunday mornings.

In 1989 I came to Leuven to study engineering and I remember how the calculus courses taught by Robert Piessens in his characteristic lively way stimulated me. In those days, however, an introduction to complex analysis was not part of the first two years of the engineering curriculum as it is today. Instead I had to wait for the introductory course taught by Ann Haegemans to a small group of fourth year students—those who had chosen to specialize in applied mathematics. Meanwhile I read several books on my own. However, by the time I arrived in my fourth year, the course given by Ann Haegemans had been abolished and replaced by a course taught by Robert Piessens to the second year students! During my fifth and final year of engineering, two important things happened. First, Adhemar Bultheel guided me towards my master’s thesis, in which I presented a matrix generalization of his Szegö theory for rational functions. And secondly, I followed a course given by Ann Haegemans on applications of real and complex analysis. This turned out to be a very rewarding experience. I learned a lot in this course, especially about fractional calculus and Mellin and Hankel transforms. When by the end of that year the time came to choose a topic for a PhD thesis, I knew in which direction I wanted to do research. I chose the problem of computing zeros of analytic functions and I was lucky enough to find some clues in an early paper by Delvès and Lyness and also in a book written by the Russian mathematicians Aizenberg and Yuzhalov, in which they formulated a multidimensional version of the logarithmic residue formula used by Delvès and Lyness. The first part of this thesis has grown out of these two references. Finally, during my first PhD year I took the complex analysis course given by Alfons Van Daele at the Department of Mathematics. I remember it as a very interesting, clear and didactic course.

The previous account of “how I came to write a thesis on computational complex analysis” is of course only a partial and very teleological reconstruction. Besides, this thesis contains other ingredients as well, notably structured numerical linear algebra, numerical analysis and some informatics. But I have always had the inner conviction that I want to do research on computational complex analysis and one may read this thesis with that in mind.

I sincerely thank Ann Haegemans for accepting to be my promotor. She gave me the freedom to explore my research subject on my own while stimulating me via her kind and intelligent remarks. The results on Newton’s method are joint work.

Marc Van Barel has a strong numerical intuition from which I have learned a lot. I thank him for guiding me through the world of formal orthogonal polynomials and for introducing me to structured matrices. The last part of this thesis was written in close collaboration with him.

I spent two months at the Department of Mathematics of the University of Patras (Greece). I sincerely thank Michael Vrahatis, Omiros Ragos and Filaretis Zafiropoulos for their kind hospitality. They introduced me to problems concerning zeros of Bessel functions. The package ZEBEC was written in close collaboration
with them. Michael Vrahatis mentioned certain important combinations of Bessel functions to me. However, I didn't know how to tackle this particular problem. Back in Leuven, I mentioned it casually to Pierre Verlinden who, the next day, came back to my office telling me that he had an idea that might lead us somewhere. One thing indeed led to another and within less than two weeks we found all the proofs that we were looking for. These results are presented in the second part of this thesis.

I enjoy attending conferences. This thesis brought me to Luminy (France), Cortona (Italy), Hamburg (Germany), Nicosia (Cyprus), Kuwait City (Kuwait), Madrid (Spain) and Athens (Greece). There is something unusual about meeting more or less the same group of people about once a year, each time in a different city. I thank Georg Heinig, Dario Bini, Stefano Serra, Plamen Yalamov, Bernard Mourrain, Jean-Paul Cardinal and Ioannis Emiris for many cordial and stimulating conversations.

Many other people contributed to this thesis. Ronald Cools arranged my first visit to Patras and gave me advice on various compiler and Fortran 90 issues. I thank Johan Quaeghebeur for joining the thesis committee and for the very enthusiastic way in which he showed interest in my work. Jan Verschelde introduced me to Dario Bini (during the SEA 95 workshop) and to Michael Vrahatis (during the ICCAM 96 conference). I thank Annie Cuyt for her friendly encouragements and for introducing me to Tetsuya Sakurai. Ria Vanhove is a very competent librarian. I sincerely thank her for the efficient way in which she obtained photocopies of papers for me from other libraries. I thank Philip Miller for reassuring me about my English. I shared offices with Frank Piessens, Marc Van Barel, Bart Maerten and Tom Michiels. The atmosphere was always very pleasant and creative.

This research was financed via a four year grant (1 October 1994 – 30 September 1998) from the Flemish Institute for the Promotion of Scientific and Technological Research in the Industry (IWT). Currently, I'm being paid as a research assistant at the Division of Numerical Analysis and Applied Mathematics. I thank the IWT and the K.U.Leuven for their financial support. Vlaamse Leergangen Leuven generously supported my research stays in Patras and the Fund for Scientific Research-Flanders (FWO-Vlaanderen) paid my ticket to Madrid. I thank Dario Bini for the support from the European Community that he was able to obtain for me to cover my expenses in Cortona. I thank Georg Heinig for giving me the opportunity to attend the FAA 98 conference in Kuwait. This was quite an experience. Most of my other travel and hotel expenses were financed by the FWO-Vlaanderen, projects #G.0261.96 “Counting and computing all isolated solutions of systems of nonlinear equations” and #G.0278.97 “Orthogonal Systems and their Applications.”

And, last but not least, I thank my friends for the many nice moments and I thank my father for his love and support, for the possibilities, the freedom and the independence.

Peter Kravanja

Peter.Kravanja@na-net.ornl.gov
Nederlandse samenvatting

Dit proefschrift is een mengeling van numerieke complexe analyse en numerieke lineaire algebra. Het is onderverdeeld in vier delen.

We bestuderen eerst het probleem van het berekenen van alle nulpunten van een analytische functie $f$ die binnen een positief georiënteerde Jordankromme $\gamma$ liggen. Ons belangrijkste middel om informatie over de ligging der nulpunten te bekomen, bestaat uit een symmetrische, bilineaire vorm die geëvalueerd kan worden via numerieke integratie langs $\gamma$. Deze vorm bevat de logaritmische afgeleide $f'/f$ van $f$. Onze aanpak kan bijgevolg een kwadraatuurmethode genoemd worden die gebaseerd is op logaritmische residu’s. In zekere zin zetten wij het pionierswerk van Delves en Lyness verder. We werpen een nieuw licht op hun aanpak door andere onbekenden te beschouwen en door gebruik te maken van de theorie der formele orthogonale veeltermen. Ons algoritme bereikt niet alleen benaderingen voor de nulpunten maar ook hun respectievelijke multipliciteiten. Het vereist geen initiële benaderingen voor de nulpunten en we hebben vastgesteld dat het nauwkeurige resultaten geeft. Een Fortran 90 implementatie staat ter beschikking (het pakket ZEAL). We stellen ook een aanpak voor die de afgeleide $f'$ niet vereist. Deze resultaten worden behandeld in Hoofdstuk 1.

In Hoofdstuk 2 spitsen we onze aandacht toe op het bepalen van clusters van nulpunten van analytische functies. We tonen aan hoe de aanpak die we in Hoofdstuk 1 voorgesteld hebben, gebruikt kan worden om een benadering te berekenen voor het middelpunt van een cluster en tevens het totale aantal nulpunten dat tot deze cluster behoort. We benaderen ons probleem van het berekenen van alle nulpunten van $f$ die binnen $\gamma$ liggen ook op een volledig andere manier, gebaseerd op rationale interpolatie in wortels uit één. We geven aan hoe deze nieuwe aanpak de vorige aanvult en op een efficiënte manier gebruikt kan worden indien $\gamma$ de eenheidscirkel is.

In Hoofdstuk 3 tonen we aan hoe onze op logaritmische residu’s gebaseerde aanpak gebruikt kan worden om alle nulpunten en polen van een meromorfe functie te berekenen die binnen een Jordankromme liggen.

In Hoofdstuk 4 beschouwen we stelsels analytische vergelijkingen. Een meerdimensionale logaritmische residuformule is beschikbaar in de literatuur. Deze formule bevat de integraal van een differentiaalvorm. We herschrijven deze integraal als een som van Riemann-integraal en tonen aan hoe de oplossingen en hun respectievelijke multipliciteiten uit deze integraal berekend kunnen worden door een veralgemeende eigenwaardenprobleem met Hankel-structuur op te lossen en een aantal Vandermonde-stelsels.
Dit vormt het einde van Deel 1. In Deel 2 beschouwen we analytische functies die alleen maar enkelvoudige nulpunten hebben, in het bijzonder Besselfuncties. In Hoofdstuk 5 spitsen we onze aandacht toe op de Besselfuncties $J_\nu(z)$, $Y_\nu(z)$, $H^{(1)}_\nu(z)$ en $H^{(2)}_\nu(z)$, en hun eerste afgeleiden, indien de veranderlijke $z \in \mathbb{C} \setminus [\infty, 0]$ en de orde $\nu \in \mathbb{R}$. We stellen ons softwarepakket ZEBEC voor, dat bedoeld is om alle nulpunten van deze functies te berekenen die binnen een rechthoek liggen waarvan de zijden evenwijdig zijn met de coördinaatassen. In Hoofdstuk 6 beschouwen we de functies $J_n(z) \pm i J_{n+1}(z)$ en $[J_{n+1}(z)]^2 - J_n(z)J_{n+2}(z)$ indien $n \in \mathbb{N}$. De nulpunten van deze functies spelen een belangrijke rol in bepaalde natuurkundige toepassingen. We tonen aan dat alle nulpunten in $\mathbb{C}_0$ enkelvoudig zijn en geven aan hoe ZEBEC op een eenvoudige manier aangepast kan worden om deze nulpunten te berekenen.

In Deel 3 beschouwen we de methode van Newton. In Hoofdstuk 7 stellen we een aanpassing van de methode van Newton voor om meervoudige oplossingen van stelsels analytische vergelijkingen te berekenen. Onder milde voorwaarden convergeert onze iteratie kwadratisch. Ze bevat bepaalde constanten waarvan het product een benedengrens is voor de multipliciteit van de oplossing. Vermits deze constanten meestal niet op voorhand gedaan zijn, leiden we een iteratie af waarin niet alleen een benadering voor de oplossing verfijnd wordt, maar ook benaderingen voor deze constanten.

In Deel 4 beschouwen we vragenstukken uit de gestructureerde numerieke lineaire algebra. We hebben rationale interpolatie al ontmoet op het einde van Hoofdstuk 2 waar we ervan gebruik gemaakt hebben om clusters van nulpunten van analytische functies te bepalen. In Hoofdstuk 8 beschouwen we rationale interpolatie in meer detail. We stellen gestabiliseerde snelle en supersnelle algoritmes voor rationale interpolatie in wortels uit één voor. In Hoofdstuk 9 gebruiken we deze algoritmes om gestabiliseerde snelle en supersnelle algoritmes te construeren voor het oplossen van (indefiniete) stelsels lineaire vergelijkingen met Hankel- of Toeplitz-structuur. Zo'n stelsels komen in heel wat toepassingen voor, bijv. in signaalverwerking of bij Markov-ketens. Ze spelen ook een centrale rol in de theorie der orthogonale veeltermen en bij Padé-benadering. Een Fortran 90 implementatie van onze algoritmes staat ter beschikking.

Dit proefschrift is een mengeling van theoretische resultaten (sommige daarvan zijn vrij technisch, bijv. de resultaten omtrent stelsels analytische vergelijkingen, rationale interpolatie of de methode van Newton), numerieke analyse en algorithmische aspecten, implementatieheuristieken en afgewerkte software (ZEAL, ZEBEC en onze software voor Hankel- en Toeplitz-stelsels).

Deel 1

1. Nulpunten van analytische functies

In dit hoofdstuk spitsen we onze aandacht toe op het volgende probleem. We wensen alle nulpunten van een analytische functie $f$ te berekenen die binnen een Jordankromme $\gamma$ liggen. Ons algoritme berekent niet alleen benaderingen voor de
nulpunten maar ook hun respectievelijke multipliciteiten. Het vereist geen initiële benaderingen voor de nulpunten en geeft nauwkeurige resultaten. Het is gebaseerd op de theorie der formele orthogonale veeltermen. Om informatie over de ligging der nulpunten te bekomen, maken we gebruik van een symmetrische bilineaire vorm die geëvalueerd kan worden via numerieke integratie langs $\gamma$. Deze vorm bevat de logaritmische afgeleide $f'/f$ van $f$. Onze aanpak kan dus als volgt omschreven worden: het is een kwadratuurmethode gebaseerd op logaritmische residu's.

In de volgende hoofdstukken van het eerste deel van dit proefschrift zullen we aantonen hoe een gelijkvaardige aanpak gebruikt kan worden om clusters van nulpunten van analytische functies te bepalen, om nulpunten en polen van meromorfe functies te berekenen, en om stelsels analytische vergelijkingen op te lossen.

Dit hoofdstuk stemt gedeeltelijk overeen met onze artikels [186], [188], [190] en [192].

Inleiding

Zij $W$ een enkelvoudig samenhangend gebied in $\mathbb{C}$, $f : W \to \mathbb{C}$ analytisch in $W$ en $\gamma$ een positief georiënteerde Jordankromme in $W$ die door geen enkel nulpunt van $f$ gaat. We wensen alle nulpunten van $f$ te berekenen die binnen $\gamma$ liggen, samen met hun respectievelijke multipliciteiten.

Onze aanpak is een voortzetting van het pionierswerk geleverd door Delves en Lyness [70]. Zij $N$ het totale aantal nulpunten van $f$ die in het inwendige van $\gamma$ liggen. Veronderstel vanaf nu dat $N > 0$. Delves en Lyness beschouwden de rij $Z_1, \ldots, Z_N$ bestaande uit alle nulpunten van $f$ die binnen $\gamma$ liggen. Elk nulpunt wordt hierbij zo vaak herhaald als zijn multipliciteit. Men kan eenvoudig nagaan dat de logaritmische afgeleide $f'/f$ in elk nulpunt van $f$ een enkelvoudige pool heeft met residu gelijk aan de multipliciteit van het nulpunt. De stelling van Cauchy impliceert dan dat

$$N = \frac{1}{2\pi i} \int_{\gamma} \frac{f'(z)}{f(z)} \, dz.$$  

Deze formule laat ons toe $N$ te berekenen via numerieke integratie. Methodes voor het berekenen van nulpunten van analytische functies die gebaseerd zijn op het numeriek evalueren van integralen worden kwadratuurmethodes genoemd. Een overzicht van dergelijke methodes werd gegeven door Ioakimidis [158]. Delves en Lyness beschouwden de integralen

$$s_p := \frac{1}{2\pi i} \int_{\gamma} z^p \frac{f'(z)}{f(z)} \, dz, \quad p = 0, 1, 2, \ldots.$$  

De residuensstelling impliceert dat de $s_p$'s gelijk zijn aan de Newtonsommen van de onbekende nulpunten,

$$s_p = Z_1^p + \cdots + Z_N^p, \quad p = 0, 1, 2, \ldots.$$  

In wat volgt zullen we veronderstellen dat alle vereiste $s_p$'s reeds berekend werden. In het bijzonder geldt dit voor $N = s_0$. 
Delvess en Lyness beschouwden de monische veelterm van graad $N$ met nulpunten $Z_1, \ldots, Z_N$,

$$P_N(z) := \prod_{k=1}^{N} (z - Z_k) = z^N + \sigma_1 z^{N-1} + \cdots + \sigma_N.$$ 

De coëfficiënten van $P_N(z)$ kunnen berekend worden via de identiteiten van Newton. Op deze manier herleidden ze het probleem tot het eenvoudigere probleem van het berekenen van de nulpunten van een veelterm. Helaas is de afbeelding van de Newtonssommen $s_1, \ldots, s_N$ naar de coëfficiënten $\sigma_1, \ldots, \sigma_N$ meestal slecht geconditioneerd. Daarom moeten de integraal uit nauwkeurig berekend worden indien $N$ groot is. Vandaar dat Delvess en Lyness de veelterm $P_N(z)$ slechts opstelden indien de graad kleiner is dan een zekere getal $M$. Anders wordt het inwendige van $\gamma$ onderverdeeld en worden de kleinere gebieden één na één beschouwd. Delvess en Lyness kopen $M$ gelijk aan 5.

Botten, Craig en McPhedran [41] implementeerden de methode van Delvess en Lyness in Fortran 77.

Het probleem met deze aanpak is volgens ons dat men de verkoorte onbekenden beschouwt. Men dient de onderling verschillende nulpunten en hun respectieve multipliciteiten afzonderlijk te beschouwen. Dit is de aanpak die wij zullen volgen. Zij $n$ het aantal onderling verschillende nulpunten van $f$ die binnen $\gamma$ liggen. Zij $z_1, \ldots, z_n$ deze nulpunten en $\nu_1, \ldots, \nu_n$ hun respectieve multipliciteiten. Onze kwadratuurmethode is een veralgemening van de methode van Delvess en Lyness. We zullen aantonen hoe de onderling verschillende nulpunten berekend kunnen worden door veralgemende eigenwaardenproblemen op te lossen. De waarde van $n$ zullen we op een indirecte manier bepalen. Zodra $n$ en $z_1, \ldots, z_n$ gegeven zijn, kunnen de multipliciteiten $\nu_1, \ldots, \nu_n$ berekend worden door een Vandermonde-stelsel op te lossen.

**Formele orthogonale veeltermen**

Zij $\mathcal{P}$ de vectorruimte der veeltermen met complexe coëfficiënten. We definiëren de symmetrische bilineaire vorm

$$\langle \cdot, \cdot \rangle : \mathcal{P} \times \mathcal{P} \rightarrow \mathbb{C}$$

als volgt. Zij

$$(1) \quad \langle \phi, \psi \rangle := \frac{1}{2\pi i} \int_\gamma \phi(z) \psi(z) \frac{f'(z)}{f(z)} \, dz = \sum_{k=1}^{n} \nu_k \phi(z_k) \psi(z_k)$$

voor elke twee veelterm $\phi, \psi \in \mathcal{P}$. De laatste gelijkheid volgt uit het feit dat $f'/f$ een enkelvoudige pool heeft in $z_k$ met residu $\nu_k$ voor $k = 1, \ldots, n$. Noteer dat $\langle \cdot, \cdot \rangle$ geëvalueerd kan worden via numerieke integratie langs $\gamma$. We veronderstellen vanaf nu dat alle vereiste “inwendige producten” $\langle \phi, \psi \rangle$ reeds berekend worden. Zij $s_p := \langle 1, z^p \rangle$ voor $p = 0, 1, 2, \ldots$. Deze gewone momenten zijn gelijk aan de Newtonssommen van de onbekende nulpunten,

$$s_p = \sum_{k=1}^{n} \nu_k z_k^p, \quad p = 0, 1, 2, \ldots.$$
In het bijzonder geldt dat \( s_0 = \nu_1 + \cdots + \nu_n = N \), het totale aantal nulpunten. We mogen dus veronderstellen dat de waarde van \( N \) bekend is. Zij \( H_k \) de \( k \times k \)-Hankel-matrix

\[
H_k := \begin{bmatrix} s_{p+q} & s_{p+q-1} & \cdots & s_{p+q-k+1} \\ s_{p+q-1} & \ddots & \cdots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ s_{p+q-k+1} & \cdots & \cdots & s_{p+q-2k+1} \end{bmatrix}
\]

voor \( k = 1, 2, \ldots \). Een monische veelterm \( \varphi_t \) van graad \( t \geq 0 \) die voldoet aan

\[
\langle z^k, \varphi_t(z) \rangle = 0, \quad k = 0, 1, \ldots, t-1,
\]

wordt een *formele orthogonale veelterm* (FOP) van graad \( t \) genoemd. Het adjectief *formeel* legt de nadruk op het feit dat in het algemeen de vorm \( \langle \cdot, \cdot \rangle \) geen (positief definit) inwendig product definiert. Een belangrijke gevolg hiervan is dat formele orthogonale veeltermen niet noodzakelijk voor elke graad \( t \) bestaan of enig zijn. (Voor meer details verwijzen we naar Draux [71, 72], Gutknecht [128, 130] of Gragg en Gutknecht [113].) Indien aan (2) voldaan is en \( \varphi_t \) enig is, dan wordt \( \varphi_t \) een *reguliere* FOP genoemd en \( t \) een *reguliere index*. Definieer

\[
\varphi_t(z) =: u_{0,t} z + u_{1,t} z^2 + \cdots + u_{t-1,t} z^{t-1} + z^t.
\]

Dan kan voorwaarde (2) herschreven worden als

\[
\begin{bmatrix} s_0 & s_1 & \cdots & s_{t-1} \\ s_1 & \ddots & \cdots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ s_{t-1} & \cdots & \cdots & s_{2t-2} \end{bmatrix} \begin{bmatrix} u_{0,t} \\ \vdots \\ u_{t-1,t} \end{bmatrix} = - \begin{bmatrix} s_t \\ \vdots \\ s_{2t-1} \end{bmatrix}.
\]

De reguliere FOP van graad \( t \geq 1 \) bestaat bijgevolg als en slechts als de matrix \( H_t \) regulier is.

De volgende stelling laat ons toe om, in theorie althans, \( n \) te berekenen als de rang van \( H_N \).

**Stelling.** \( n = \text{rang} H_{n+p} \) voor elk niet-negatief geheel getal \( p \). In het bijzonder is \( n \) gelijk aan de rang van \( H_N \).

Bijgevolg is \( H_n \) regulier terwijl \( H_t \) singulier is voor \( t > n \). Noteer dat \( H_1 = [s_0] \) regulier is bij veronderstelling. De reguliere FOP van graad \( 1 \) bestaat en is gelijk aan \( \varphi_1(z) = z - \mu \) waarbij

\[
\mu := \frac{s_1}{s_0} = \frac{\sum_{k=1}^{n} \nu_k z_k}{\sum_{k=1}^{n} \nu_k}
\]

het rekenkundige gemiddelde van de nulpunten is. De vorige stelling impliceert dat de reguliere FOP \( \varphi_n \) van graad \( n \) bestaat en dat er geen reguliere FOP's zijn van graad groter dan \( n \). Men kan eenvoudig nagaan dat \( \varphi_n \) gelijk is aan

\[
\varphi_n(z) = (z - z_1) \cdots (z - z_n).
\]

Het is de monische veelterm van graad \( n \) die \( z_1, \ldots, z_n \) als enkelvoudige nulpunten heeft.
Zodra de waarde van $n$ bekend is, kunnen de onderling verschillende nullpunten $z_1, \ldots, z_n$ berekend worden door een veralgemeend eigenwaardenprobleem op te lossen. Zij $H_n^\leq$ de Hankel-matrix

$$
H_n^{\leq} := \begin{bmatrix}
  s_1 & s_2 & \cdots & s_n \\
  s_2 & \ddots & \ddots & \vdots \\
  \vdots & \ddots & \ddots & \vdots \\
  s_n & \cdots & \cdots & s_{2n-1}
\end{bmatrix}.
$$

**Stelling.** De eigenwaarden van de pencil $H_n^{\leq} - \lambda H_n$ zijn gelijk aan $z_1, \ldots, z_n$.

Zodra $n$ en $z_1, \ldots, z_n$ bekend zijn, kunnen de multipliciteiten $\nu_1, \ldots, \nu_n$ berekend worden door het Vandermonde-stelsel

$$
\begin{bmatrix}
  1 & \cdots & 1 \\
  z_1 & \cdots & z_n \\
  \vdots & \ddots & \vdots \\
  z_1^{n-1} & \cdots & z_n^{n-1}
\end{bmatrix}
\begin{bmatrix}
  \nu_1 \\
  \nu_2 \\
  \vdots \\
  \nu_n
\end{bmatrix}
= \begin{bmatrix}
  s_0 \\
  s_1 \\
  \vdots \\
  s_{n-1}
\end{bmatrix},
$$

op te lossen.

De vorige stellingen suggereren dus de volgende aanpak om $n$ en $z_1, \ldots, z_n$ te bepalen. Eerst berekenen we het totale aantal nullpunten $N$. Vervolgens berekenen we $s_1, \ldots, s_{2N-2}$. We hebben reeds vermeld dat dit kan gebeuren met behulp van numerieke integratie langs $\gamma$. Het aantal onderling verschillende nullpunten berekenen we als de rang van $H_N$. De nullpunten $z_1, \ldots, z_n$ bekomen we tenslotte door een veralgemeend eigenwaardenprobleem op te lossen. Deze aanpak heeft echter verschillende nadelen:

- In theorie zijn de $N - n$ kleinste singuliere waarden van $H_N$ gelijk aan nul. Bij numerieke berekeningen zal dit echter niet het geval zijn en is het dus moeilijk om de rang van $H_N$ en dus de waarde van $n$ te bepalen indien de kloof tussen de berekende benaderingen voor de singuliere waarden die in theorie gelijk zijn aan nul en de singuliere waarden die in theorie verschillend zijn van nul te klein is.

- De bekomen benaderingen voor $z_1, \ldots, z_n$ kunnen zeer onnauwkeurig zijn. De afbeelding van de Newtonsommen naar de nullpunten en hun respectieve multipliciteiten

$$
(s_0, s_1, \ldots, s_{2n-1}) \mapsto (z_1, \ldots, z_n, \nu_1, \ldots, \nu_n)
$$

is immers meestal zeer slecht geconditioneerd. We verwijzen in dit verband ook naar de artikels van Gautschi [93, 96, 97] die de conditie van (5) onderzocht in verband met Gauss kwadratuurformules.

Vandaar dat we op zoek gaan naar een algoritme dat tot nauwkeurigere benaderingen van $z_1, \ldots, z_n$ leidt. De achterliggende idee is de volgende. De inwendige producten in de Hankel-matrices $H_n$ en $H_n^{\leq}$ hebben te maken met de klassieke basis van de machten van $z$. Waarom beschouwen we geen andere basis? Het feit dat $H_n$ en $H_n^{\leq}$
kunnen geschreven worden als
\[
H_n = \left[ \langle z^p, z^q \rangle \right]_{p,q=0}^{n-1} \quad \text{en} \quad H_n^\perp = \left[ \langle z^p, z z^q \rangle \right]_{p,q=0}^{n-1}
\]
suggereert dat we matrizen van de vorm
\[
\left[ \langle \psi_p, \psi_q \rangle \right]_{p,q=0}^{n-1} \quad \text{en} \quad \left[ \langle \psi_p, \psi_1 \psi_q \rangle \right]_{p,q=0}^{n-1}
\]
beschouwen, met \( \psi_k \) een veelterm van graad \( k \) voor \( k = 0, 1, \ldots, n - 1 \). Het is zo dat, zelfs indien we erin slagen om de vorige stelling te formuleren in termen van deze nieuwe matrizen, we op die manier nog geen antwoord gevonden hebben op de vraag welke veeltermen \( \psi_k \) we best zouden gebruiken. We hebben vastgesteld dat we zeer nauwkeurige resultaten bekomen indien we de formele orthogonaliteitseigenschappen van FOP’s. Met andere woorden, de nulpunten van \( \varphi_n(z) \) zullen bekomen uit inwendige producten die te maken hebben met de veeltermen \( \varphi_0(z), \varphi_1(z), \ldots, \varphi_{n-1}(z) \). Vooralere we dit in detail kunnen uitleggen, moeten we echter eerst iets dieper ingaan op de orthogonaliteitseigenschappen van FOP’s.

Indien \( H_n \) sterk regulier is, i.e., indien alle leidende principale deelmatrices van \( H_n \) regulier zijn, dan bestaat de volledige veramzelling \( \{ \varphi_0, \varphi_1, \ldots, \varphi_n \} \) van reguliere FOP’s. Wat gebeurt er indien \( H_n \) niet sterk regulier is? Door de ontdrekkende reguliere FOP’s op de juiste manier te definíren, kan men een rij \( \{ \varphi_t \}_{t=0}^{\infty} \) bekomen, met \( \varphi_t \) een monische veelterm van graad \( t \), die de volgende eigenschap heeft. Indien deze veeltermen in blokken gegroepeerd worden, waarbij elk blok begint met een reguliere FOP en verder geen reguliere FOP’s bevat, dan zijn de veeltermen die tot verschillende blokken behoren onderling orthogonal met betrekking tot \( \langle \cdot, \cdot \rangle \). De rij \( \{ \varphi_t \}_{t=0}^{\infty} \) wordt als volgt gedefiníren. Indien \( t \) een reguliere index is, dan definíren we \( \varphi_t \) als de reguliere FOP van graad \( t \). Anders definíren we \( \varphi_t \) als \( \varphi_r \psi_s, r \) waarbij \( r \) de grootste reguliere index is die kleiner is dan \( t \) en \( \psi_s, r \) een willekeurige monische veelterm van graad \( t-r \) is. In dit geval noemen we \( \varphi_t \) een inwendige veelterm. Deze veeltermen \( \{ \varphi_t \}_{t=0}^{\infty} \) kunnen in blokken gegroepeerd worden. Elk blok begint met een reguliere FOP en bevat verder alleen maar inwendige veeltermen. Uit de blok orthogonaliteitseigenschap volgt dan dat de Gram matrix \( G_n := [\langle \varphi_r, \varphi_s \rangle]_{r,s=0}^{n-1} \) blok diagonal is. De diagonaalblokken zijn regulier, symmetrisch en nul boven de hoofdantidiagonaal. (Voor meer details verwijzen we naar Bultheel en Van Barel [47].)

De vorige stelling kan als volgt gedefiníren worden: de nulpunten van de reguliere FOP van graad \( n \) kunnen berekend worden door een veralgemeend eigenwaardenprobleem op te lossen. De volgende stelling toont aan dat dit voor alle reguliere FOP’s geldt. Dit zal ons toelaten FOP’s te berekenen in hun productvorm. Definíeer de matrizen \( G_{k} \) en \( G_{k}^{(1)} \) als
\[
G_k := \left[ \langle \varphi_r, \varphi_s \rangle \right]_{r,s=0}^{k-1} \quad \text{en} \quad G_k^{(1)} := \left[ \langle \varphi_r, \varphi_1 \varphi_s \rangle \right]_{r,s=0}^{k-1}
\]
voor \( k = 1, 2, \ldots, \). 

**Stelling.** Zij \( t \geq 1 \) een reguliere index en zij \( z_{i,1}, \ldots, z_{i,t} \) de nulpunten van de reguliere FOP \( \varphi_t \). Dan zijn de eigenwaarden van de pencil \( G_t^{(1)} - \lambda G_t \) gelijk aan \( \varphi_1(z_{i,1}), \ldots, \varphi_t(z_{i,t}) \). Met andere woorden, ze zijn gelijk aan \( z_{i,1} - \mu, \ldots, z_{i,t} - \mu \) waarbij \( \mu = s_1/s_0 \).
Stelling. De eigenwaarden van $G_n^{(1)} - \lambda G_n$ zijn gelijk aan $z_1 - \mu, \ldots, z_n - \mu$ waarbij $\mu = s_1/s_0$.

Reguliere FOP’s worden gekenmerkt door het feit dat de determinant van een Hankel-matrix verschillend is van nul, terwijl inwendige veeltermen overeenkomen met singuliere Hankel-matrices. Om te beslissen of $\varphi_t$ als een reguliere FOP of als een inwendige veelterm berekend moet worden, zou men dus de determinant van $H_t$ kunnen berekenen en vergelijken met nul. Bij numerieke berekeningen heeft een dergelijke test “is gelijk aan nul” echter geen zin. Omwille van afrondingsfouten (zowel bij het berekenen van $\langle \cdot, \cdot \rangle$ als bij het berekenen van de determinant) zou men alleen maar reguliere FOP’s ontmoeten. Strikt genomen zou men kunnen beweren dat inwendige veeltermen eigenlijk overbodig zijn bij numerieke berekeningen. Het tegendeel is waar! Laten we een reguliere FOP goed geconditioneerd noemen indien het overeenkomstige stelsel (3) goed geconditioneerd is. In het andere geval spreken we van een slecht geconditioneerde reguliere FOP. Om een numeriek stabiel algoritme te bekomen, is het cruciaal om enkel goed geconditioneerde reguliere FOP’s te genereren en slecht geconditioneerde reguliere FOP’s te vervangen door inwendige veeltermen.

Ons algoritme voor het berekenen van de onderling verschillende nulpunten $z_1, \ldots, z_n$ berekent achtereenvolgens de veeltermen $\varphi_0(z), \varphi_1(z), \ldots, \varphi_n(z)$ in hun productvorm, te beginnen met $\varphi_0(z) \leftarrow 1$ en $\varphi_1(z) \leftarrow z - \mu$. Het algoritme maakt gebruik van een heuristiek om te beslissen of de volgende veelterm in de rij al dan niet als een reguliere FOP berekend moet worden. We hebben een groot aantal numerieke experimenten uitgevoerd en zijn op die manier tot de conclusie gekomen dat onze heuristiek tot nauwkeurige resultaten leidt. Voor meer details verwijzen we naar de Engelse tekst.

De volgende stelling laat ons toe de waarde van $n$ te bepalen.

**Stelling.** Zij $t \geq n$. Dan is $\langle z^p, \varphi_t(z) \rangle = 0$ voor alle $p \geq 0$.

Veronderstel dat ons algoritme net een (goed geconditioneerde) reguliere FOP $\varphi_r(z)$ gegenereerd heeft. Om te verifiëren of $n = r$, doorloopt het algoritme de rij

$$\left(\left|\langle z^r, \varphi_r(z) \rangle \right|^2\right)^{N-1-r}_{r=0}.$$  

Indien alle elementen “voldoende klein” zijn, dan besluit het algoritme dat inderdaad $n = r$ en het stopt.

We hebben reeds vermeld dat zodra $n$ en (benaderingen voor) $z_1, \ldots, z_n$ gekend zijn, de multipliciteiten $\nu_1, \ldots, \nu_n$ bepaald kunnen worden door het Vandermonde-stelsel (4) op te lossen. We hebben ons algoritme in Matlab geïmplementeerd. In de Engelse tekst illustreren we onze aanpak met behulp van een aantal numerieke voorbeelden.

**Het softwarepakket ZEAL**

We hebben ons algoritme geïmplementeerd voor het geval dat de kromme $\gamma$ een rechthoek is waarvan de zijden evenwijdig zijn met de coördinaatassen. Ons pakket heet ZEAL (‘ZEros of AnaLytic functions’) en is geschreven in Fortran 90. Numerieke benaderingen voor de integralen langs $\gamma$ worden berekend met behulp van het kwadratuurpakket QUADPACK [241].
Voor meer details omtrent de structuur van ZEAL, de in- en uitvoerparameters en een aantal numerieke voorbeelden verwijzen we naar de Engelstalige tekst.

Een aanpak die de afgeleide \( f' \) niet vereist

De resultaten die we tot nu toe besproken hebben, zijn gebaseerd op de bilineaire vorm (1). Deze vorm bevat de logaritmische afgeleide \( f'^{0} = f \). Zij

\[
\langle \cdot, \cdot \rangle_{*} : \mathcal{P} \times \mathcal{P} \to \mathbb{C}
\]

de symmetrische bilineaire vorm gedefinieerd als

\[
\langle \phi, \psi \rangle_{*} := \frac{1}{2\pi i} \int_{\gamma} \phi(z) \overline{\psi(z)} \frac{1}{f(z)} \, dz
\]

voor elke twee veeltermen \( \phi, \psi \in \mathcal{P} \). We tonen aan dat met behulp van deze vorm essentieel dezelfde resultaten behaald kunnen worden als voorheen met \( \langle \cdot, \cdot \rangle \). De afgeleide \( f' \) is niet vereist. Het spreekt vanzelf dat we niet de onderling verschillende nulpunten zullen kunnen berekenen maar slechts de onbekenden \( Z_{1}, \ldots, Z_{N} \) (zoals gedefinieerd in de inleiding). De multipliciteiten kunnen niet expliciet berekend worden. Afgezien daarvan geeft deze nieuwe aanpak echter vergelijkbare resultaten. We illustreren dit aan de hand van een aantal numerieke voorbeelden.

2. Clusters van nulpunten van analytische functies

De technieken die we in het vorige hoofdstuk voorgesteld hebben, kunnen ook gebruikt worden om clusters van nulpunten van analytische functies te bepalen, meer bepaald het zwartepunt van een cluster en het aantal nulpunten dat tot een cluster behoort. Dit hoofdstuk stemt overeen met ons artikel [186].

Veronderstel dat de nulpunten van \( f \) die binnen \( \gamma \) liggen, gegroepeerd kunnen worden in \( m \) clusters. Zij \( I_{1}, \ldots, I_{m} \) indexverzamelingen die deze clusters definiëren, en zij

\[
\mu_{j} := \sum_{k \in I_{j}} \nu_{k} \quad \text{en} \quad c_{j} := \frac{1}{\mu_{j}} \sum_{k \in I_{j}} \nu_{k} z_{k}
\]

voor \( j = 1, \ldots, m \). Met andere woorden, \( \mu_{j} \) is gelijk aan het totale aantal nulpunten die cluster \( j \) vormen (het “gewicht” van de cluster) terwijl \( c_{j} \) het rekenkundige gemiddelde is van de nulpunten uit cluster \( j \) (het “zwartepunt” van de cluster). We veronderstellen dat de zwartepunten \( c_{1}, \ldots, c_{m} \) onderling verschillend zijn. Voor \( k = 1, \ldots, n \) definieren we ook \( \zeta_{k} := z_{k} - c_{j} \) indien \( k \in I_{j} \). We definiëren de symmetrische bilineaire vorm \( \langle \cdot, \cdot \rangle_{m} \) als volgt:

\[
\langle \phi, \psi \rangle_{m} := \sum_{j=1}^{m} \mu_{j} \phi(c_{j}) \overline{\psi(c_{j})}
\]

voor elke twee veeltermen \( \phi, \psi \in \mathcal{P} \). Definieer

\[
\delta := \max_{1 \leq k \leq n} |\zeta_{k}|
\]

De volgende stelling leert ons dat de vormen \( \langle \cdot, \cdot \rangle_{m} \) en \( \langle \cdot, \cdot \rangle \) elkaar benaderen.
De volgende stelling zal ons toelaten benaderingen voor de zwartepunten te bekomen.

Stelling. Zij $t$ een geheel getal $\geq m$. Dan geldt dat $\varphi_t(c_j) = \mathcal{O}(\delta^2)$, $\delta \to 0$ voor $j = 1, \ldots, m$. Ook geldt dat $\langle z^p, \varphi_t(z) \rangle = \mathcal{O}(\delta^2)$, $\delta \to 0$ voor alle $p \geq t$.

Met andere woorden, wellicht zullen we, tenzij de FOP $\varphi_t(z)$ een zeer vlak verloop heeft in de buurt van zijn nullpunten, goede benaderingen voor de zwartepunten $c_1, \ldots, c_m$ aantreffen tussen de nullpunten van $\varphi_t(z)$ en dit voor alle $t \geq m$. Noteer dat

$$
\begin{bmatrix}
1 & \cdots & 1 \\
1 & \cdots & c_m \\
\vdots & \vdots & \vdots \\
c_1 & \cdots & c_m \\
\end{bmatrix}
\begin{bmatrix}
\mu_1 \\
\mu_2 \\
\vdots \\
\mu_m \\
\end{bmatrix}
= 
\begin{bmatrix}
0 \\
\delta_1 \\
\vdots \\
\delta_{m-1} \\
\end{bmatrix} + \mathcal{O}(\delta^2), \delta \to 0.
$$

Bijgevolg kunnen we benaderingen voor de gewichten $\mu_1, \ldots, \mu_m$ bekomen door een Vandermonde-stelsel op te lossen.

In de Engelse tekst zullen we in detail illustreren hoe deze stellingen ons toelaten om met behulp van het algoritme dat we in het vorige hoofdstuk voorgesteld hebben clusters te bepalen. We zullen ook een equivalentie aanpak bespreken die gebaseerd is op rationale interpolatie.

3. Nulpunten en polen van meromorfe functies

Veronderstel nu dat $f$ niet analytisch maar meromorf is in $W$ en dat $f$ nulpunten noch polen heeft op $\partial W$. Zij $P$ het totale aantal polen van $f$ die binnen $\gamma$ liggen. Veronderstel dat $N + P > 0$. Zij $p$ het aantal onderling verschillende polen van $f$ die binnen $\gamma$ liggen. Zij $y_1, \ldots, y_p$ deze polen en $\mu_1, \ldots, \mu_p$ hun respectievelijke ordes. Men kan eenvoudig nagaan dat $f/f$ een eenvoudige pool heeft in $y_i$ met residu $-\mu_i$ voor $i = 1, \ldots, p$. Bijgevolg geldt dat

$$
\langle \phi, \psi \rangle = \sum_{k=1}^n \nu_k \phi(z_k) \psi(z_k) - \sum_{i=1}^p \mu_i \phi(y_i) \psi(y_i).
$$

Deze uitdrukking is van dezelfde vorm als (1). De resultaten behaald in het eerste hoofdstuk maken geen gebruik van het feit dat de multipliciteiten $\nu_k$ posities gehele getallen zijn. Ze steunen enkel op het feit dat $\nu_k \neq 0$ voor $k = 1, \ldots, n$. We tonen aan dat, op voorwaarde dat een bovengrens voor $P$ gekend is, ons algoritme uit het eerste hoofdstuk vrij eenvoudig aangepast kan worden om alle nulpunten en polen van $f$ te berekenen die binnen $\gamma$ liggen.

Dit hoofdstuk stemt overeen met ons artikel [191].
4. Stelsels analytische vergelijkingen

In dit hoofdstuk tonen we aan hoe de logaritmische residu-technieken uit het eerst hoofdstuk gebruikt kunnen worden om stelsels analytische vergelijkingen op te lossen. Dit hoofdstuk stemt overeen met ons artikel [183].

Zij \( d \geq 1 \) een positief geheel getal. Beschouw een polyschijf \( D \in \mathbb{C}^d \) (i.e., \( D \) is het Cartesische product van \( d \) schijven in \( \mathbb{C} \)) en zij \( f = (f_1, \ldots, f_d): \overline{D} \to \mathbb{C}^d \) een afbeelding die analytisch is in \( \overline{D} \) en geen nulpunten heeft op de rand van \( D \). We wensen alle nulpunten van \( f \) te berekenen die in \( D \) liggen, samen met hun respectievelijke multipliciteit. Zij \( Z_f(D) \) de verzameling der nulpunten van \( f \) die in \( D \) liggen en zij \( \mu_a(f) \) de multipliciteit van een nulpunt \( a \in Z_f(D) \).

Indien \( d = 1 \), dan leert ons de klassieke formule die we in de vorige hoofdstukken gebruikt hebben dat

\[
\frac{1}{2\pi i} \int_{\partial D} \varphi(z) \frac{f'(z)}{f(z)} \, dz = \sum_{a \in Z_1(D)} \mu_a(f) \varphi(a)
\]

indien \( \varphi: \overline{D} \to \mathbb{C} \) analytisch is in \( D \) en continu in \( \overline{D} \).

De volgende stelling is een meerdimensionale versie van dit resultaat. Zij \( J_f \) de Jacobiaanmatrix van \( f \) en zij \( J_{[k]} \) de Jacobiaanmatrix van \( f \) met de \( k \)de kolom vervangen door \([f_1 \ldots f_d]^T\):

\[
J_{[k]} := \begin{bmatrix}
\frac{\partial f_1}{\partial z_1} & \cdots & \frac{\partial f_1}{\partial z_d} \\
\vdots & \ddots & \vdots \\
\frac{\partial f_d}{\partial z_1} & \cdots & \frac{\partial f_d}{\partial z_d}
\end{bmatrix} \qquad k = 1, \ldots, d
\]

Veronderstel dat de polyschijf \( D \) kan geschreven worden als

\[
D = D_1 \times \cdots \times D_d
\]

waarbij

\[
D_k = \{ z \in \mathbb{C} : |z - C_k| < R_k \}, \quad k = 1, \ldots, d,
\]

met \( C_1, \ldots, C_d \in \mathbb{C} \) en \( R_1, \ldots, R_d > 0 \).

**Stelling.** Zij \( \varphi: \overline{D} \to \mathbb{C} \) analytisch in \( D \) en continu in \( \overline{D} \). Definieer \( I_k(\varphi) \) voor \( k = 1, \ldots, d \) als de integraal

\[
I_k(\varphi) := \rho_k \int_{[0,1]^{d-1}} \frac{\varphi(z_1, \ldots, z_d) \det J_f(z_1, \ldots, z_d) \det J_{[k]}(z_1, \ldots, z_d)}{\left( |f_1(z_1, \ldots, z_d)|^2 + \cdots + |f_d(z_1, \ldots, z_d)|^2 \right)^d} \times e^{2\pi i \theta_1 r_1 \cdots r_{k-1} r_{k+1} \cdots r_d} dr_1 \cdots dr_{k-1} dr_{k+1} \cdots dr_d \, d\theta_1 \cdots d\theta_d
\]

met

\[
\rho_k = \rho(d, R_1, \ldots, R_d; k) := 2^{d-1} (d-1)! R_1^2 \cdots R_{k-1}^2 R_k R_{k+1}^2 \cdots R_d^2
\]

en waarbij

\[
z_k = z_k(\theta_k) = C_k + R_k e^{2\pi i \theta_k} \quad 0 \leq \theta_k \leq 1
\]

en

\[
z_l = z_l(r_l, \theta_l) = C_l + r_l R_l e^{2\pi i \theta_l} \quad 0 \leq r_l, \theta_l \leq 1
\]
voor \( l \in \{1, \ldots, d\} \setminus \{k\} \). Dan geldt dat

\[
I(\varphi) := \sum_{k=1}^{d} I_k(\varphi) = \sum_{a \in \mathbb{Z}_l(D)} \mu_a(f) \varphi(a).
\]

De stelling zal ons toelaten om met behulp van numerieke integratie informatie te bekomen over de ligging van de nulpunten van \( f \). Op basis van deze informatie kunnen deze nulpunten en hun respectievelijke multipliciteiten berekend worden door een veralgemeende eigenwaardenprobleem op te lossen en \( d \) Vandermonde-stelsels. Voor meer details verwijzen we naar de Engelse tekst.

Deel 2

5. Het berekenen van enkelvoudige nulpunten van Besselfuncties

In dit hoofdstuk vestigen we onze aandacht op het probleem van het berekenen van enkelvoudige nulpunten van analytische functies. In het bijzonder beschouwen we de Besselfuncties \( J_\nu(z) \), \( Y_\nu(z) \), \( H^{(1)}_\nu(z) \), \( H^{(2)}_\nu(z) \) en hun eerste afgeleide, waarbij de veranderlijke \( z \) tot \( \mathbb{C} \setminus (-\infty, 0) \) behoort en de orde \( \nu \) reëel is. We stellen de lezer ons softwarepakket ZEBEC voor, waarmee voor elk van deze functies alle nulpunten berekend kunnen worden die binnen een rechtthoek liggen waarvan de zijden evenwijdig met de coördinaatassen zijn.

Dit hoofdstuk stemt overeen met ons artikel [185].

Inleiding

De Besselvergelijking

\[
z^2 u''(z) + z u'(z) + (z^2 - \nu^2) u(z) = 0, \quad z, \nu \in \mathbb{C},
\]

verschijnt in heel wat vraagstukken uit de wiskundige natuurkunde. De functies \( J_\nu(z) \), de Besselfunctie van de eerste soort en van orde \( \nu \), en \( Y_\nu(z) \), de Besselfunctie van de tweede soort en van orde \( \nu \), zijn twee lineair onafhankelijke oplossingen van deze vergelijking. Ook van belang zijn de Besselfuncties van de derde soort,

\[
H^{(1)}_\nu(z) := J_\nu(z) + i Y_\nu(z) \quad \text{en} \quad H^{(2)}_\nu(z) := J_\nu(z) - i Y_\nu(z).
\]

Deze functies zijn analytisch in \( \mathbb{C} \setminus (-\infty, 0] \).

De nulpunten en keerpunten (i.e. nulpunten van de eerste afgeleide) van Besselfuncties spelen een belangrijke rol in de natuurkunde en de ingenieurswetenschappen. Ze komen voor bij de studie van de trillingen van een membraan, de temperatuurverdeling in een cilinder of een bol, de diffractie van een vlakke elektromagnetische golf door een geleidende cilinder, enz. Voor een elektrostatische interpretatie van de nulpunten van Besselfuncties verwijzen we naar [220].

In [240, 242] werd een softwarepakket voorgesteld voor het berekenen van nulpunten van \( J_\nu(x) \) indien \( x > 0 \) en \( \nu > -1 \) en keerpunten van \( J_\nu(x) \) indien \( x > 0 \) en \( \nu > 0 \). Het pakket RFSFNS [291] kan gebruikt worden om nulpunten en keerpunten van \( J_\nu(x) \) en \( Y_\nu(x) \) te berekenen indien \( x > 0 \) en \( \nu \geq 0 \). Onlangs hebben Segura
en Gil [253] twee algoritmes voorgesteld om reële nulpunten van $J_\nu(x)$ te berekenen indien de orde $\nu$ reëel is (zowel positief als negatief). In dit hoofdstuk stellen we een softwarepakket voor om nulpunten of keerpunten van $J_\nu(z)$, $Y_\nu(z)$, $H^{(1)}_\nu(z)$ en $H^{(2)}_\nu(z)$ te berekenen indien de veranderlijke $z$ tot $\mathbb{C}\setminus(-\infty,0]$ behoort en de orde $\nu$ reëel is. Ons softwarepakket heet ZEBEC: ‘ZEros of BEssel functions Complex.’ Het kan alle nulpunten of keerpunten berekenen die in een rechthoek liggen waarvan de zijden evenwijdig zijn met de coördinaatassen. Elke oplossing van (6) heeft alleen maar enkelvoudige nulpunten, tenzij eventueel in het punt $z = 0$ [295, p. 479]. Dit geldt in het bijzonder voor $J_\nu(z)$, $Y_\nu(z)$, $H^{(1)}_\nu(z)$ en $H^{(2)}_\nu(z)$. De afgeleide van een oplossing van (6) heeft ook slechts enkelvoudige nulpunten, tenzij eventueel in $z = 0$ of $z = \pm \nu$ [175, 176, 267]. De nulpunten die we wensen te berekenen zijn dus allemaal enkelvoudig, tenzij eventueel in $z = 0$ en $z = \pm \nu$. Vermits we veronderstellen dat het argument $z$ tot $\mathbb{C}\setminus(-\infty,0]$ behoort en de orde $\nu$ reëel is, hoeven we $z = 0$ en één van de punten $z = \pm \nu$ niet te beschouwen. Door de functie en haar eerste afgeleide te evalueren in $z = \nu$ (indien $\nu > 0$) of $z = -\nu$ (indien $\nu < 0$), kan men cyaatvoudig nagaan of de functie in dit punt een meervoudig nulpunt heeft of niet. Indien dit het geval is, dan waarschuwt ZEBEC de gebruiker en vraagt om een ander rechthoekig gebied. In het andere geval beginnen de berekeningen.

### Enkelvoudige nulpunten van analytische functies

De nulpunten van Besselfuncties die we wensen te berekenen, zijn dus allen enkelvoudig. Vandaar dat we eerst het volgende, iets algemener probleem beschouwen. Zij $f$ een analytische functie met alleen maar enkelvoudige nulpunten. Om alle nulpunten van $f$ te berekenen die binnen een rechthoek liggen waarvan de zijden evenwijdig zijn met de coördinaatassen gaan we als volgt te werk:

1. We berekenen het totale aantal nulpunten die binnen deze rechthoek liggen.
2. We verdelen deze rechthoek onder in deelrechthoecken tot we alle nulpunten geïsoleerd hebben. Op die manier bekomen we een verzameling rechthoecken die elk precies één nulpunt bevatten.
3. We berekenen de nulpunten die in elk van deze rechthoecken liggen.

Tijdens de eerste en de tweede fase maken we gebruik van de logaritmische residu-integraal

$$\frac{1}{2\pi i} \int_C f'(z) f(z) \, dz.$$ 

Hierbij is de kromme $C$ een rechthoek en we tonen aan hoe in dit geval deze integraal geschreven kan worden als een som van vier Riemann-integraal. Deze integraal benaderen we met behulp van het integratiepakket QUADPACK [241]. Tijdens de derde fase maken we gebruik van het pakket CHABIS [288, 289]. Dit pakket bevat een implementatie van een veralgemeende bisectionmethode die karakteristieke bisection genoemd wordt. Het is gebaseerd op de theorie van de topologische graad.

### Het softwarepakket ZEBEC

Ons pakket ZEBEC is gebaseerd op het vorige algemene schema voor het berekenen van enkelvoudige nulpunten van analytische functies. Om de Besselfuncties te evalueren maken we gebruik van het pakket BESSCC [268]. Voor een gedetailleerde
beschrijving van de structuur en de invoerparameters van het pakket verwijzen we naar de Engelse tekst. We illustreren hoe ZEBEC gebruikt kan worden om alle nulpunten van $Y_{15.3}(z)$ te berekenen die gelegen zijn in het rechthoekige gebied

$$ \{ z \in \mathbb{C} : -22 \leq \text{Re} z \leq 23, \quad 0.5 \leq \text{Im} z \leq 100.5 \}. $$

6. De nulpunten van $J_n(z) \pm iJ_{n+1}(z)$ en $[J_{n+1}(z)]^2 - J_n(z)J_{n+2}(z)$

In dit hoofdstuk beschouwen we de complexe nulpunten van de functies

$$ z \mapsto J_n(z) \pm iJ_{n+1}(z) $$

en

$$ z \mapsto [J_{n+1}(z)]^2 - J_n(z)J_{n+2}(z) $$

waarbij $n \in \mathbb{N}$ en $J_n(z)$ de Besselfunctie van de eerste soort en orde $n$ voorstelt. Definieer $F_n(z) := J_n(z) - iJ_{n+1}(z)$ en $G_n(z) := [J_{n+1}(z)]^2 - J_n(z)J_{n+2}(z)$.

De nulpunten van deze functies spelen een belangrijke rol in bepaalde fysische toepassingen. De nulpunten van $F_0(z)$ zijn van belang bij de studie van watergolven die een hellend strand ontmoeten, cf. Synolakis [263, 264]. De vergelijking $F_n(z) = 0$ verschijnt bij stranden met meerdere hellingen [266]. MacDonald [205] gebruikte de nulpunten van $G_0(z)$ om representatieve stroomlijnen te tekenen voor de stationaire stroming van een viskleuze vloeiostof in een lange buis met constante straal die rond haar as (de $\xi$-as) draait met een hoek snelheid die bij $\xi = 0$ op een discontinue manier verandert van de ene constante waarde naar een andere constante waarde met hetzelfde teken.

MacDonald [204, 205, 206] leidde asymptotische formules af voor de nulpunten van $F_0(z)$, $F_n(z)$ en $G_n(z)$. Deze formules laten toe nulpunten te benaderen die een grote modulus hebben. MacDonald stelde vast dat de nauwkeurigheid van de asymptotische formules voor de nulpunten van $F_0(z)$ vermindert naarmate $n$ toeneemt. Een ander nadeel van deze asymptotische formules is dat ze vaak zeer onnauwkeurige resultaten geven voor kleinere nulpunten. Deze moeten dan via een of andere numerieke procedure berekend worden. MacDonald [206] ging uit van een afgebroken machtreeks voor $F_0(z)$ en berekende de nulpunten daarvan via de methode van Newton.

We stellen voor om de nulpunten van $F_n(z)$ en $G_n(z)$ te berekenen met behulp van ons softwarepakket ZEBEC. Op de eerste plaats is dit pakket ontworpen om nulpunten van de Besselfuncties $J_n(z)$, $Y_n(z)$, $H_n^{(1)}(z)$, $H_n^{(2)}(z)$ en hun eerste afgeleide te berekenen, maar het pakket kan eenvoudig aangepast worden om nulpunten van een willekeurige analytische functie te berekenen, op voorwaarde natuurlijk dat deze nulpunten onbekend zijn. In $z = 0$ hebben $F_n(z)$ en $G_n(z)$ een nulpunt met multipliciteit $n$ (indien $n \geq 1$) en $2n + 2$, respectievelijk. We bewijzen dat alle nulpunten in $\mathbb{C}_0$ onbekend zijn. Ook tonen we aan dat er geen nulpunten op de coördinatenassen liggen (behalve dan in $z = 0$). Ter illustratie gebruiken we ZEBEC om de eerste 30 nulpunten van $J_5(z) - iJ_6(z)$ te berekenen die in het vierde kwadrant gelegen zijn.

Dit hoofdstuk stemt overeen met ons artikel [193].
Deel 3

7. De methode van Newton voor meervoudige nulpunten

In dit hoofdstuk stellen we een aanpassing van de methode van Newton voor, specifiek voor het berekenen van meervoudige oplossingen van stelsels analytische vergelijkingen. Onze iteratie convergeert kwadratisch onder milde voorwaarden. Ze bevat bepaalde constanten waarvan het product een benedengrens vormt voor de multiplicité van het nulpunt. Deze constanten zijn zelden op voorhand gekend. Vandaar dat we een iteratieschema opstellen waarin niet alleen een benadering voor het nulpunt verfijnd wordt, maar ook benaderingen voor deze constanten.

Dit hoofdstuk stemt overeen met ons artikel [184].

Inleiding

Zij \( f : \mathbb{C} \rightarrow \mathbb{C} \) een zachte functie en veronderstel dat \( z^* \) een nulpunt van \( f \) is met multiplicité \( \mu \). Indien \( \mu = 1 \), dan convergeert de methode van Newton kwadratisch naar \( z^* \) indien de startwaarde van de iteratie voldoende dicht in de buurt van \( z^* \) ligt. Indien \( \mu > 1 \), dan convergeert de iteratie slechts lineair. Indien \( \mu \) gekend is, dan kan men opnieuw kwadratische convergentie bekomen via de iteratie

\[
(7) \quad z^{(p+1)} = z^{(p)} - \mu \frac{f(z^{(p)})}{f'(z^{(p)})}, \quad p = 0, 1, 2, \ldots
\]

Van de Vel [281, 283] ontwierp een iteratie waarin niet alleen een benadering voor het nulpunt verfijnd wordt, maar ook een schatting van de multiplicité. King [177] analyseerde de methode van Van de Vel en toonde aan dat de convergentieradius gelijk is aan 1.554. Hij herschikte de volgorde van de berekeningen en bekwam op die manier een iteratie met convergentieradius gelijk aan 1.618. Zijn iteratieschema ziet er als volgt uit:

\[
(8) \quad \left\{ \begin{array}{l}
\mu^{(p+1)} = \frac{u(z^{(p)})}{u(z^{(p)}) - u(z^{(p+1)})}\mu^{(p)} \\
z^{(p+2)} = z^{(p+1)} - \mu^{(p+1)}u(z^{(p+1)})
\end{array} \right.
\]

voor \( p = 0, 1, 2, \ldots \), waarbij \( u(0) \) en \( \mu(0) \) gegeven zijn en \( z^{(1)} := z^{(0)} - \mu^{(0)}u(z^{(0)}) \).

Hierbij is \( u(z) := f(z)/f'(z) \).

We veralgemenen deze resultaten naar meerdere dimensies. We beschouwen stelsels analytische vergelijkingen, i.e., analytische afbeeldingen \( f : \mathbb{C}^n \rightarrow \mathbb{C}^n \).

Een wijziging van de methode van Newton

Zij \( f = f(z) : \mathbb{C}^n \rightarrow \mathbb{C}^n \) een analytische afbeelding, waarbij \( z = (z_1, \ldots, z_n) \) en \( f = (f_1, \ldots, f_n) \). Een punt \( z^* \in \mathbb{C}^n \) heet een nulpunt van \( f \) indien \( f(z^*) = 0 \). Een geïsoleerd nulpunt \( z^* \) van \( f \) heet enkelvoudig indien de Jacobiaanmatrix van \( f \) regulier is in \( z^* \). De volgende stelling is overgenomen uit het bekende boek van Aizenberg en Yuzhakov [3].
STELLING. Indien de sluiting van een omgeving $U_\varepsilon$ van een nulpunt $z^*$ van $f$ geen ander nulpunt van $f$ bevat, dan bestaat er een $\varepsilon > 0$ zodanig dat voor bijna elke $\zeta \in \mathbb{C}^*$, $\|\zeta\|_2 < \varepsilon$, de afbeelding

$$z \mapsto f(z) - \zeta$$

alleen maar enkelvoudige nulpunten heeft in $U_\varepsilon$. Het aantal nulpunten hangt niet af van de keuze van $\zeta$ of van de omgeving $U_\varepsilon$.

Het aantal nulpunten van de afbeelding (9) die in de omgeving $U_\varepsilon$ liggen, wordt de multipliciteit van $z^*$ genoemd en genoteerd als $\mu_{z^*}(f)$. De multipliciteit van een geïsoleerd nulpunt van een analytische afbeelding is dus gelijk aan het aantal enkelvoudige nulpunten waarin dit nulpunt uiteenvalt onder een voldoende kleine perturbatie van de afbeelding.

Zij nu $z^* = (z_1^*, \ldots, z_n^*)$ een geïsoleerd nulpunt van $f = (f_1, \ldots, f_n)$ zodanig dat

$$f_j(z) = \sum_{|\alpha| \geq k_j} c_{j,\alpha} (z - z^*)^\alpha$$

voor $j = 1, \ldots, n$ waarbij $\alpha$ een multi-index is, $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$, $|\alpha| = \alpha_1 + \cdots + \alpha_n$ en $(z - z^*)^\alpha = (z_1 - z_1^*)^{\alpha_1} \cdots (z_n - z_n^*)^{\alpha_n}$. We noemen $k_j$ de orde van $z^*$ als nulpunt van $f_j$. Definieer

$$P_j(z) := \sum_{|\alpha| = k_j} c_{j,\alpha} (z - z^*)^\alpha$$

voor $j = 1, \ldots, n$. De homogene polynomiale afbeelding

$$P = P(z) := (P_1(z), \ldots, P_n(z))$$

wordt het homogene principale deel van $f$ in $z^*$ genoemd. De volgende stelling van Tsikh en Yuzhakov legt een verband tussen de multipliciteit $\mu_{z^*}(f)$ en de ordes $k_1, \ldots, k_n$.

STELLING. De multipliciteit van een geïsoleerd nulpunt $z^*$ van $f$ is gelijk aan het product van de ordes van $z^*$ als nulpunt van $f_1, \ldots, f_n$ als en slechts als $z^*$ een geïsoleerd nulpunt is van de afbeelding (10). Er geldt dat $\mu_{z^*}(f) \geq k_1 \cdots k_n$.

De volgende stelling is onze veralgemening van iteratie (7).

STELLING. Veronderstel dat $z^{(0)}$ zodanig is dat de iteratie

$$z^{(p+1)} = z^{(p)} - [f'(z^{(p)})]^{-1} \text{diag}(k_1, \ldots, k_n)f(z^{(p)}), \quad p = 0, 1, 2, \ldots,$$

goed gedefinieerd is voor elke $p$. Indien dit $P'(z) \neq 0$ en indien $z^{(0)}$ voldoende dicht in de buurt van $z^*$ ligt, dan convergeert $z^{(p)}$ kwadratisch naar $z^*$. Indien dit $P'(z) \equiv 0$, dan is de convergentie slechts lineair.

In de Engelse tekst geven we een aantal numerieke voorbeelden om deze stelling te illustreren.
De methode van Van de Vel

De iteratie uit de vorige stelling kan slechts toegepast worden indien de ordes $k_1, \ldots, k_n$ gekend zijn. We leiden een iteratieschema af waarbij zowel een benadering voor het nulpunt als benaderingen voor de ordes verfijnd worden. De benaderingen voor de ordes $k_1, \ldots, k_n$ in de $p$de iteratiestap noteren we als $d_1^{(p)}, \ldots, d_n^{(p)}$. Zij

$$d^{(p)} := \begin{bmatrix} d_1^{(p)} \\ \vdots \\ d_n^{(p)} \end{bmatrix}$$

en definieer de matrixfunctie

$$U(z) := [f'(z)]^{-1} \text{diag}(f_1(z), \ldots, f_n(z))$$

voor elke $z \in \mathbb{C}^n$ waarvoor $f'(z)$ regulier is. Dan ziet onze veralgemening van (8) er als volgt uit:

$$d^{(p+1)} = U(z^{(p)}) - U(z^{(p)} - U(z^{(p)}d^{(p)}))^{-1} U(z^{(p)}d^{(p)})$$
$$z^{(p+1)} = (z^{(p)} - U(z^{(p)}d^{(p)})) - U(z^{(p)} - U(z^{(p)}d^{(p)}))d^{(p+1)}$$

voor $p = 0, 1, 2, \ldots$, gegeven een schatting $z^{(0)}$ voor $z^*$ en schattingen $d^{(0)}$ voor de ordes. Dit is onze meerdimensionale versie van de methode van Van de Vel. We illustreren deze iteratiemethode met behulp van een aantal voorbeelden. Deze suggereren dat de convergente dezelfde is als in het eendimensionale geval.

Deel 4

8. Supersnelle rationele interpolatie

We hebben rationele interpolatie al ontmoet op het einde van Hoofdstuk 2 bij het bepalen van clusters van nulpunten van analytische functies. In dit hoofdstuk gaan we dieper in op de problematiek van het oplossen van rationele interpolatieproblemen. De meeste gekende algoritmes zijn numeriek onstabiel. We spitsen onze aandacht toe op het algoritme van Van Barel en Bultheel [274]. We tonen aan hoe dit algoritme gestabiliseerd kan worden via pivoting. Met behulp van een verdeel-en-heers strategie bekomen we ook een recursief algoritme dat we stabiliseren door zogenaamde “moeilijke interpolatiepunten” een speciale behandeling te geven, met behulp van iteratieve verfijning gebaseerd op de inverse van een gekoppelde Vandermonde-matrix en met behulp van “downdating.”

Dit hoofdstuk is vrij technisch van aard. Het is bedoeld om samen met het volgende hoofdstuk gelezen te worden, waarin we onze interpolatieroutines zullen gebruiken bij het oplossen van stelsels lineaire vergelijkingen die Hankel- of Toeplitz-structuur hebben. Dit hoofdstuk is gedeeltelijk terug te vinden in onze artikels [187], [189], [278] en [279].
Het algoritme RatInt

Zij \( n \) een positief geheel getal. Veronderstel dat \( s_1, \ldots, s_{2n} \in \mathbb{C} \) en \( f_1, \ldots, f_{2n} \in \mathbb{C}^{2 \times 1} \) gegeven zijn. Beschouw het interpolatieprobleem
\[
\begin{bmatrix}
f_k^T B(s_k) = [0 \ 0], & k = 1, \ldots, 2n,
\end{bmatrix}
\]
waarbij
\[
B(z) = \begin{bmatrix} n_d(z) & n_r(z) \\ d_d(z) & d_r(z) \end{bmatrix} \in \mathbb{C}[z]^{2 \times 2}
\]
met \( \deg n_d(z) = n \), \( \deg d_d(z) \leq n - 1 \), \( \deg n_r(z) \leq n - 1 \) en \( \deg d_r(z) = n \), en zowel \( n_d(z) \) als \( d_d(z) \) monisch, i.e. waarbij \( B(z) \) een monische \( 2 \times 2 \) matrixveelterm van graad \( n \) is. Veronderstel dat dit interpolatieprobleem precies één oplossing \( B^*(z) \) heeft. In het volgende hoofdstuk zullen we rationale interpolatieproblemen ontmoeten die eenvoudig op deze manier geformuleerd kunnen worden.

Ons algoritme RatInt berekent een \( 2 \times 2 \) matrixveelterm \( B(z) \) van graad \( n \) die aan (11) voldoet en die een reguliere hoogstegraadscoëfficiënt \( A \in \mathbb{C}^{2 \times 2} \) heeft. Dit impliceert dat \( B(z) \equiv B^*(z)A \). We tonen aan dat dit \( A = 1 \). Ons algoritme is gebaseerd op het algoritme van Van Barel en Bultheel. Het maakt gebruik van pivotering om de numerieke stabiliteit te verbeteren. Het vereist \( O(n^2) \) flops en behoort dus tot de categorie der snelle algoritmes. Supersnelle algoritmes vereisen slechts \( O(n \log^2 n) \) flops.

We geven ook een matrixversie van RatInt. In dit geval behoren de vectoren \( f_k \) tot \( \mathbb{C}^{p \times 1} \) waarbij \( p \) een positief geheel getal is. Voor meer details verwijzen we naar de Engelse tekst.

Een gestabiliseerde verdeel-en-heers aanpak

Ons algoritme RatInt gaat sequentieel te werk: de interpolatiepunten worden één na één afgehandeld. (Pivotering zorgt ervoor dat de volgorde der interpolatiepunten.eventueel omgewisseld wordt.) In plaats van alle interpolatiepunten in één keer te beschouwen, kan men de verzameling interpolatiepunten echter ook in twee (gelijke) delen opsplitsen. De overeenkomstige interpolatieproblemen kunnen dan afzonderlijk opgelost worden. We tonen aan hoe beide oplossingen samengevoegd kunnen worden tot de oplossing van het oorspronkelijke probleem. Dit schema kan recursief toegepast worden. De interpolatieproblemen op het allerlaagste niveau (met het kleinste aantal interpolatiepunten) worden opgelost met behulp van het snelle algoritme RatInt. Dit leidt tot een supersnel algoritme dat we stabiliseren door de technieken die we hoger reeds vermeld hebben.

9. Supersnelle algoritmes voor het oplossen van stelsels lineaire vergelijkingen met Hankel- of Toeplitz-structuur

Hankel- en Toeplitz-matrices komen in heel wat toepassingen voor, bijvoorbeeld bij signaalverwerking of Markov-ketens [34, 213]. Ze spelen ook een centrale rol in de theorie der formele orthogonale veeltermen en bij Padé-benadering. We hebben Hankel-matrices al ontmoet in het eerste deel van dit proefschrift. In dit hoofdstuk stellen we snelle en supersnelle algoritmes voor om stelsels lineaire vergelijkingen
met Hankel- of Toeplitz-structuur op te lossen. ‘Snel’ betekent dat een \( n \times n \) stelsel opgelost wordt met \( O(n^2) \) flops. Een ‘supersnel’ algoritme vereist \( O(n \log^2 n) \) flops.

Dit hoofdstuk is gedeeltelijk terug te vinden in onze artikels [187], [189], [278] en [279].

**Stelsels met Hankel-structuur**

Zij \( n \) een positief geheel getal, zij \( H = H_n := [h_{k+l}]_{k,l=0}^{n-1} \) een reguliere complexe \( n \times n \) Hankel-matrix en zij \( b \in \mathbb{C}^n \). We spitsen onze aandacht toe op het berekenen van \( x := H^{-1} b \).

**Transformatie naar een stelsel met Loewner-structuur**

Zij \( y_1, \ldots, y_n, z_1, \ldots, z_n \) onderling verschillende complexe getallen en definieer \( y := (y_1, \ldots, y_n) \) en \( z := (z_1, \ldots, z_n) \). Zij \( L(y, z) \) de verzameling matrices

\[
L(y, z) := \{ \left[ \frac{c_k - d_l}{y_k - z_l} \right]_{k,l=1}^n \mid c_1, \ldots, c_n, d_1, \ldots, d_n \in \mathbb{C} \}.
\]

De elementen van \( L(y, z) \) worden *Loewner-matrices* genoemd. Fiedler [79] toonde aan hoe Hankel-matrices in Loewner-matrices getransformeerd kunnen worden en omgekeerd. Om dit resultaat te kunnen formuleren, moeten we eerst nog enkele notaties invoeren.

Zij \( t_1, \ldots, t_n \) complexe getallen en definieer \( t := (t_1, \ldots, t_n) \).

Zij \( f_t(z) \) de monische veeterm van graad \( n \) met nullpunten \( t_1, \ldots, t_n \),

\[
f_t(z) := (z - t_1) \cdots (z - t_n),
\]

en definieer

\[
f_{t,j}(z) := \prod_{k \neq j} (z - t_k), \quad j = 1, \ldots, n.
\]

Noteer dat \( f_{t,j}(z) \) een monische veeterm van graad \( n - 1 \) is voor \( j = 1, \ldots, n \).

Definieer de \( n \times n \) matrix \( W(t) \) met behulp van de vergelijking

\[
\begin{bmatrix}
  f_{t,1}(z) \\
  f_{t,2}(z) \\
  \vdots \\
  f_{t,n}(z)
\end{bmatrix}
= W(t) \begin{bmatrix}
  1 \\
  z \\
  \vdots \\
  z^{n-1}
\end{bmatrix}.
\]

De \( j \)de rij van \( W(t) \) bevat dus de coefficienten van \( f_{t,j}(z) \) in termen van de basis \( \{1, z, \ldots, z^{n-1}\} \).

**Stelling.** De matrix \( L := W(y) H [W(z)]^T \) is een Loewner-matrix uit \( L(y, z) \). De parameters \( c_1, \ldots, c_n, d_1, \ldots, d_n \) zijn (op een willekeurige constante \( \xi \in \mathbb{C} \) na) gelijk aan

\[
\begin{bmatrix}
  c_1 \\
  \vdots \\
  c_n \\
  d_1 \\
  \vdots \\
  d_n
\end{bmatrix}
= W(y, z) \begin{bmatrix}
  h_0 \\
  h_1 \\
  \vdots \\
  h_{2n-2} \\
  \xi
\end{bmatrix}.
\]
Zij $\omega := \exp(2\pi i/n)$ en $\zeta := \exp(\pi i/n)$. Veronderstel nu dat de punten $y$ en $z$ gedefinieerd worden als

$$y = (1, \omega, \ldots, \omega^{n-1}) \quad \text{en} \quad z = (\zeta, \zeta \omega, \ldots, \zeta \omega^{n-1}).$$

Dan kan het Hankel-stelsel $Hx = b$ in $O(n \log n)$ flops getransformeerd worden naar een Loewner-stelsel $Lx' = b'$. Voor meer details verwijzen we naar de Engelse tekst. Het condiegetal verandert hierbij niet: $\kappa_2(L) = \kappa_2(H).

Een inversieformule voor Loewner-matrices

Van Barel en Vavrín [280, 284] hebben een expliciete formule afgeleid voor de inverse van een Loewner-matrix. Deze formule bevat een aantal parameters die berekend kunnen worden door twee rationele interpolatieproblemen op te lossen. Zo dra deze parameters gekend zijn, kan de oplossing van het Loewner-stelsel berekend worden als $x' = L^{-1}b'$. Dit vereist slechts $O(n \log n)$ flops.

Snelle en supersnelle algoritmes voor het oplossen van Hankel-stelsels

Door deze interpolatieproblemen op te lossen met behulp van RATINT bekomen we een snel algoritme voor het oplossen van Hankel-stelsels. Deze aanpak vereist $6n^2 + O(n)$ vermenigvuldigingen en evenveel optellingen. In onze numerieke voorbeelden beschouwen we verschillende types Hankel-matrices. We laten $n$ toenemen tot 10000.

We leiden ook een matrixversie af van deze transformatie-aanpak. Met behulp van de matrixversie van RATINT bekomen we op die manier een snel algoritme voor het oplossen van blok Hankel-stelsels.

In plaats van RATINT kunnen we ook gebruik maken van ons supersnel verdeel-en-heers algoritme. Dit leidt dan tot een supersnel en gestabiliseerd algoritme voor het oplossen van Hankel-stelsels, weliswaar enkel voor stelsels waarvoor $n$ een macht van 2 is. In onze numerieke voorbeelden laten we $n$ toenemen tot $2^{17} = 131072$.

Stelsels met Toeplitz-structuur

In de plaats van Hankel-stelsels beschouwen we nu Toeplitz-stelsels. We maken opnieuw gebruik van een expliciete formule voor de inverse. Deze formule bevat bepaalde veeltermen die we weerom berekennen door rationele interpolatieproblemen op te lossen. Hiervoor doen we beroep op ons supersnel verdeel-en-heers algoritme. Dit leidt dan tot een supersnel en gestabiliseerd algoritme voor het oplossen van Toeplitz-stelsels. De orde $n$ hoeft geen macht van 2 te zijn. In onze numerieke voorbeelden laten we $n$ toenemen tot $2^{18} = 262144$. 
Outline of the thesis

This thesis is a blend of computational complex analysis and numerical linear algebra. It is divided into four parts.

We start by studying the problem of computing all the zeros of an analytic function $f$ that lie inside a positively oriented Jordan curve $\gamma$. Our principal means of obtaining information about the location of the zeros is a certain symmetric bilinear form that can be evaluated via numerical integration along $\gamma$. This form involves the logarithmic derivative $f'/f$ of $f$. Our approach could therefore be called a logarithmic residue based quadrature method. It can be seen as a continuation of the pioneering work done by Delves and Lyness. We shed new light on their approach by considering a different set of unknowns and by using the theory of formal orthogonal polynomials. Our algorithm computes not only approximations for the zeros but also their respective multiplicities. It does not require initial approximations for the zeros and we have found that it gives accurate results. The algorithm proceeds by solving generalized eigenvalue problems and a Vandermonde system. A Fortran 90 implementation is available (the package ZEAL). We also present an approach that uses only $f$ and not its first derivative $f'$. These results are presented in Chapter 1.

In Chapter 2 we focus on the problem of locating clusters of zeros of analytic functions. We show how the approach presented in Chapter 1 can be used to compute approximations for the centre of a cluster and the total number of zeros in this cluster. We also attack the problem of computing all the zeros of $f$ that lie inside $\gamma$ in an entirely different way, based on rational interpolation at roots of unity. We show how the new approach complements the previous one and how it can be used effectively in case $\gamma$ is the unit circle.

In Chapter 3 we show how our logarithmic residue based approach can be used to compute all the zeros and poles of a meromorphic function that lie in the interior of a Jordan curve.

In Chapter 4 we consider systems of analytic equations. A multidimensional logarithmic residue formula is available in the literature. This formula involves the integral of a differential form. We transform it into a sum of Riemann integrals and show how the zeros and their respective multiplicities can be computed from these integrals by solving a generalized eigenvalue problem that has Hankel structure and by solving several Vandermonde systems.

This concludes Part 1. In Part 2 we consider analytic functions whose zeros are known to be simple, in particular certain Bessel functions. In Chapter 5 we focus on the Bessel functions $J_\nu(z), Y_\nu(z), H^{(1)}_\nu(z)$ and $H^{(2)}_\nu(z)$, and their first derivative, in case the argument $z \in \mathbb{C} \setminus (\infty, 0]$ and the order $\nu \in \mathbb{R}$. We present a software
package, called ZEBEC, for computing all the zeros of one of these functions that lie inside a rectangle whose edges are parallel to the coordinate axes. In Chapter 6 we consider the functions $J_n(z) \pm i J_{n+1}(z)$ and $[J_{n+1}(z)]^2 - J_n(z)J_{n+2}(z)$ in case $n \in \mathbb{N}$. The zeros of these functions play an important role in certain physical applications. We prove that all the zeros that lie in $\mathbb{C}_0$ are simple and show how ZEBEC can be easily extended to compute these zeros.

In Part 3 we focus on Newton’s method. In Chapter 7 we propose a modification of Newton’s method for computing multiple zeros of analytic mappings (in other words, multiple roots of systems of analytic equations). Under mild assumptions our iteration converges quadratically. It involves certain constants whose product is a lower bound for the multiplicity of the zero. As these constants are usually not known in advance, we devise an iteration in which not only an approximation for the zero is refined, but also approximations for these constants.

In Part 4 we consider problems of structured numerical linear algebra. We have already encountered rational interpolation at the end of Chapter 2 where it was used to locate clusters of zeros of analytic functions. In Chapter 8 we consider rational interpolation in much more detail. We present stabilized fast and superfast algorithms for rational interpolation at roots of unity. In Chapter 9 we use these algorithms to construct stabilized fast as well as superfast solvers for (indefinite) linear systems of equations that have Hankel or Toeplitz structure. Such linear systems occur in many applications, for example in signal processing or Markov chains. They also play a central role in the theory of orthogonal polynomials and Padé approximation. A Fortran 90 implementation of our algorithms is available.

Our work is a mixture of theoretical results (some of which are quite technical, e.g., the results on systems of analytic equations, rational interpolation or Newton’s method), numerical analysis and algorithmic aspects, implementation heuristics, and polished software that is publicly available (ZEAL, ZEBEC and our packages for solving Hankel or Toeplitz systems).
Part 1
CHAPTER 1

Zeros of analytic functions

In this chapter we will consider the problem of computing all the zeros of an analytic function \( f \) that lie in the interior of a Jordan curve \( \gamma \). The algorithm that we will present computes not only approximations for the zeros but also their respective multiplicities. It doesn’t require initial approximations for the zeros and gives accurate results. The algorithm is based on the theory of formal orthogonal polynomials. Its principal means of obtaining information about the location of the zeros is a certain symmetric bilinear form that can be evaluated via numerical integration along \( \gamma \). This form involves the logarithmic derivative \( f'/f \) of \( f \). Our approach could therefore be called a logarithmic residue based quadrature method.

In the next chapters we will see how it can be used to locate clusters of zeros of analytic functions, to compute all the zeros and poles of a meromorphic function that lie in the interior of a Jordan curve, and to solve systems of analytic equations.

Part of this chapter is contained in our papers [186], [188], [190] and [192].

1. Introduction

Let \( W \) be a simply connected region in \( \mathbb{C} \), \( f : W \to \mathbb{C} \) analytic in \( W \) and \( \gamma \) a positively oriented Jordan curve in \( W \) that does not pass through any zero of \( f \). We consider the problem of computing all the zeros of \( f \) that lie in the interior of \( \gamma \), together with their respective multiplicities.

Our approach to this problem can be seen as a continuation of the pioneering work by Delves and Lyness [70]. Let \( N \) denote the total number of zeros of \( f \) that lie in the interior of \( \gamma \), i.e., the number of zeros where each zero is counted according to its multiplicity. Suppose from now on that \( N > 0 \). Delves and Lyness considered the sequence \( Z_1, \ldots, Z_N \) that consists of all the zeros of \( f \) that lie inside \( \gamma \). Each zero is repeated according to its multiplicity. An easy calculation shows that the logarithmic derivative \( f'/f \) has a simple pole at each zero of \( f \) with residue equal to the multiplicity of the zero. Cauchy’s Theorem implies that

\[
N = \frac{1}{2\pi i} \int_{\gamma} \frac{f'(z)}{f(z)} \, dz.
\]

This formula enables us to calculate \( N \) via numerical integration. Methods for the determination of zeros of analytic functions that are based on the numerical evaluation of integrals are called quadrature methods. A review of such methods was given by Ioakimidis [158]. Delves and Lyness considered the integrals

\[
s_p := \frac{1}{2\pi i} \int_{\gamma} z^p \frac{f'(z)}{f(z)} \, dz, \quad p = 0, 1, 2, \ldots.
\]
The residue theorem implies that the $s_p$'s are equal to the Newton sums of the unknown zeros,

$$s_p = Z_1^p + \cdots + Z_N^p, \quad p = 0, 1, 2, \ldots$$

In what follows we will assume that all the $s_p$'s that are needed have been calculated. In particular, we will assume that the value of $N = s_0$ is known.

Delves and Lyness considered the monic polynomial of degree $N$ that has zeros $Z_1, \ldots, Z_N$,

$$P_N(z) := \prod_{k=1}^N (z - Z_k) = z^N + \sigma_1 z^{N-1} + \cdots + \sigma_N.$$

They called $P_N(z)$ the associated polynomial for the interior of $\gamma$. Its coefficients can be calculated via Newton’s identities.

**Theorem 1 (Newton’s identities).**

\[
s_1 + \sigma_1 = 0 \\
 s_2 + s_1 \sigma_1 + 2\sigma_2 = 0 \\
 \vdots \\
 s_N + s_{N-1} \sigma_1 + \cdots + s_1 \sigma_{N-1} + N \sigma_N = 0.
\]

**Proof.** An elegant proof was given by Carpenter and Dos Santos [54].

In this way they reduced the problem to the easier problem of computing the zeros of a polynomial. Unfortunately, the map from the Newton sums $s_1, \ldots, s_N$ to the coefficients $\sigma_1, \ldots, \sigma_N$ is usually ill-conditioned. Also, the polynomials that arise in practice may be such that small changes in the coefficients produce much larger changes in some of the zeros. This ill-conditioning of the map between the coefficients of a polynomial and its zeros was investigated by Wilkinson [300]. The location of the zeros determines their sensitivity to perturbations of the coefficients. Multiple zeros and very close zeros are extremely sensitive, but even a succession of moderately close zeros can result in severe ill-conditioning. Wilkinson states that ill-conditioning in polynomials cannot be overcome without, at some stage of the computation, resorting to high precision arithmetic.

If $f$ has many zeros in the interior of $\gamma$, then the associated polynomial is of high degree and could be very ill-conditioned. Therefore, if $N$ is large, one has to calculate the coefficients $\sigma_1, \ldots, \sigma_N$, and thus the integrals $s_1, \ldots, s_N$, very accurately. To avoid the use of high precision arithmetic and to reduce the number of integrand evaluations needed to approximate the $s_p$’s, Delves and Lyness suggested to construct and solve the associated polynomial only if its degree is smaller than or equal to a preassigned number $M$. Otherwise, the interior of $\gamma$ is subdivided or covered with a finite covering and the smaller regions are treated in turn. The choice of $M$ involves a trade-off. If $M$ is increased, then fewer regions have to be scanned. However, if $M$ is chosen too large, then the resulting associated polynomial may be ill-conditioned. Delves and Lyness chose $M = 5$.

Botten, Craig and McPhedran [41] made a Fortran 77 implementation of the method of Delves and Lyness.
In some applications, the calculation of the derivative $f'$ is more time-consuming than that of $f$. Delvès and Lyness used an integration by parts to derive a formula for $s_p$ that depends only on a multi-valued logarithm of $f$ and not on $f'$. To apply this formula, they had to keep track of the sheet on which $\log f(z)$ lies as $z$ runs along the curve $\gamma$. Unfortunately, in most cases it is impossible to do this in a completely reliable way, i.e., without accidentally overlooking any sheets. Carpentier and Dos Santos [54] and Davies [63] derived similar formulae. See also Ioakimidis and Anastassolou [161].

Instead of using Newton's identities to construct the associated polynomial, Li [199] considered (13) as a system of polynomial equations. He used a homotopy continuation method to solve this system.

What is wrong with these approaches, in our opinion, is that they consider the wrong set of unknowns. One should consider the mutually distinct zeros and their respective multiplicities separately. This is the approach that we will follow. Let $n$ denote the number of mutually distinct zeros of $f$ that lie inside $\gamma$. Let $z_1, \ldots, z_n$ be these zeros and $\nu_1, \ldots, \nu_n$ their respective multiplicities. The quadrature method that we will present generalizes the approach of Delvès and Lyness. We will show how the mutually distinct zeros can be calculated by solving generalized eigenvalue problems. The value of $n$ will be determined indirectly. Once $n$ and $z_1, \ldots, z_n$ have been found, the problem becomes linear and the multiplicities $\nu_1, \ldots, \nu_n$ can be computed by solving a Vandermonde system.

The rest of this chapter is organized as follows. In the remainder of this section we will discuss how the total number of zeros can be calculated with certainty and we will give an overview of other approaches that were proposed for computing zeros of analytic functions. In Section 2 we will tackle our problem by using the theory of formal orthogonal polynomials. This section is devoted to theoretical considerations whereas our numerical algorithm will be presented in Section 3. We have found that this algorithm gives very accurate results. In Section 4 we will give numerical examples computed via a Matlab implementation whereas in Section 5 we will present a Fortran 90 implementation (the package ZEAL) and we will give more numerical examples. In Section 6 we will present a derivative-free approach that involves $1/f$ instead of $f'/f$ and we will compare both approaches.

### 1.1. Computing the total number of zeros with certainty.

The total number of zeros of $f$ that lie inside $\gamma$ is given by the integral

$$N = \frac{1}{2\pi i} \int_{\gamma} \frac{f'(z)}{f(z)} \, dz,$$

cf. Equation (12). By making the substitution $w := f(z)$ we obtain that

$$N = \frac{1}{2\pi i} \int_{f(\gamma)} \frac{1}{w} \, dw.$$

Here $f(\gamma)$ denotes the image of the curve $\gamma$ under $f$. This is a closed curve that avoids the origin. The winding number of $f(\gamma)$ with respect to the origin is defined as the increase in the argument of $f(z)$ along $\gamma$ divided by $2\pi$, 

$$n(f(\gamma), 0) := \frac{1}{2\pi} [\arg f(z)]_{z \in \gamma}. $$
Informally speaking, one can say that it is equal to the number of times that the curve \( f(\gamma) \) “winds” itself around the origin. A classical theorem in complex analysis (see, e.g., Henrici [141, p. 233]) says that this winding number can be expressed as the integral that appears in the right-hand side of (15). Hence \( N = n(f(\gamma), 0) \). This result is known as the “principle of the argument.”

**Example 1.** Suppose that \( f(z) = e^{3z} + 20z \cos z - 1 \). Figure 1 shows the curve \( f(\gamma) \) in case \( \gamma = \{z \in \mathbb{C} : |z| = 2\} \). Clearly \( N = 4 \).

![Figure 1](image_url)

**Figure 1.** The curve \( f(\gamma) \) where \( f(z) = e^{3z} + 20z \cos z - 1 \) and \( \gamma = \{z \in \mathbb{C} : |z| = 2\} \).

If \( \gamma = \{z \in \mathbb{C} : |z - 1| = 1/2\} \), then \( N = 0 \), as can be seen from Figure 2.

Earlier in this chapter we have suggested to calculate the total number of zeros \( N \) via numerical integration, i.e., by using (14). An alternative approach can be based on an algorithm for computing winding numbers. The range of the function \( \text{arg} \) is \((-\pi, \pi]\). If the increase in argument along the straight section

\[
[\alpha, \beta] := \{z \in \mathbb{C} : z = t\alpha + (1-t)\beta, \quad 0 \leq t \leq 1 \}, \quad \alpha, \beta \in \mathbb{C},
\]

satisfies

\[
|[\text{arg} f(z)]_{z \in [\alpha, \beta]}| \leq \pi,
\]

then

\[
[\text{arg} f(z)]_{z \in [\alpha, \beta]} = \text{arg}\left(\frac{f(\beta)}{f(\alpha)}\right),
\]
as the reader may easily verify. Let us discretize the curve $\gamma$ into the sequence of points $c_1, \ldots, c_G$. Define $c_{G+1} := c_1$. Then it follows that

$$N = \frac{1}{2\pi} \sum_{k=1}^{G} \arg \left( \frac{f(c_{k+1})}{f(c_k)} \right)$$

if

$$\left| \arg f(z) \right|_{z \in [c_k, c_{k+1}]} \leq \pi$$

for $k = 1, \ldots, G$. In other words, if condition (16) is satisfied, then $N$ can be computed simply by evaluating $f$ at the points $c_1, \ldots, c_G$. These considerations form the basis for Henrici’s algorithm [141, pp. 239–241]. See also Ying and Katz [307].

Unfortunately, condition (16) may not be easy to verify for an arbitrary analytic function $f$. If the discretization of $\gamma$ is inadequate, then the computed value of $n(f(\gamma), 0)$ and hence $N$ may be wrong. In this sense Henrici’s algorithm is unreliable and the same holds for numerical integration. Indeed, a finite number of functional or derivative values are not enough to determine the number of zeros of $f$, even if $f$ is a polynomial. This was shown by Ying in his PhD thesis [306]. The next theorem is a slightly modified version of his result.

**Theorem 2.** Let $\Omega$ be a simply connected region in $\mathbb{C}$. Let $l \geq 1$ be a positive integer and let $\zeta, \ldots, \zeta_l \in \Omega$. Let $m_1, \ldots, m_l \geq 0$ be nonnegative integers and suppose that numerical values are given for

$$p(\zeta), p'(\zeta), \ldots, p^{(m_1)}(\zeta), \ldots, p(\zeta_l), p'(\zeta_l), \ldots, p^{(m_l)}(\zeta_l)$$

where $p$ is only known to be a polynomial in $\Omega$ that is not identically zero in $\Omega$. Then the following holds:
1. The information in (17) is insufficient to determine the location of all the zeros of \( p \) that lie in \( \Omega \).

Let \( \Gamma \) be a Jordan curve in \( \Omega \) that does not pass through any zero of \( p \).

2. Suppose that \( N \), the total number of zeros of \( p \) that lie inside \( \Gamma \), is known. 

   Even then, the information in (17) is insufficient to determine the location of all the zeros of \( p \) that lie inside \( \Gamma \), unless \( N = 0 \) or \( N > 0 \) and these zeros coincide with a subset of \( \{ \zeta_1, \ldots, \zeta_l \} \).

3. If the points \( \zeta_1, \ldots, \zeta_l \) do not belong to the closure of the interior of \( \Gamma \), then the information in (17) is insufficient to estimate the location of all the zeros of \( p \) that lie inside \( \Gamma \).

Proof.

1. Define \( h(z) \) as \((z - \zeta_1)^{m_1+1} \cdots (z - \zeta_l)^{m_l+1} \). Choose \( \zeta_0 \) in \( \Omega \setminus \{ \zeta_1, \ldots, \zeta_l \} \) such that \( p(\zeta_0) \neq 0 \) and let

   \[
   p_1(z) := p(z) - \frac{p(\zeta_0)}{h(\zeta_0)} h(z).
   \]

   Then \( p_1 \) is a polynomial in \( \Omega \) and \( p_1^{(k)}(\zeta_i) = p^{(k)}(\zeta_i) \) for \( i = 1, \ldots, l \) and \( k = 0, 1, \ldots, m_i \). However, \( p_1(\zeta_0) = 0 \) while \( p(\zeta_0) \neq 0 \).

2. Define

   \[
   \epsilon := \inf_{z \in \Gamma} |p(z)| \quad \text{and} \quad M := \sup_{z \in \Gamma} |h(z)|.
   \]

   Then \( \epsilon > 0 \) and \( M > 0 \). Let

   \[
   p_2(z) := p(z) + \frac{\epsilon}{2M} h(z).
   \]

   Then \( p_2 \) is a polynomial in \( \Omega \) and \( p_2^{(k)}(\zeta_i) = p^{(k)}(\zeta_i) \) for \( i = 1, \ldots, l \) and \( k = 0, 1, \ldots, m_i \). Since \( |p(z)| > \frac{\epsilon}{2M} |h(z)| \) for all \( z \in \Gamma \), Rouche’s Theorem asserts that \( p_2 \) and \( p \) have the same total number of zeros in the interior of \( \Gamma \). Let \( \zeta \) be a zero of \( p \) that lies in the interior of \( \Gamma \). If \( \zeta \not\in \{ \zeta_1, \ldots, \zeta_l \} \), then \( p_2(\zeta) \neq 0 \).

3. Define

   \[
   \delta := \inf_{z \in \Gamma} |h(z)| \quad \text{and} \quad Q := \sup_{z \in \Gamma} |p(z)|.
   \]

   Then \( \delta > 0 \) and \( Q > 0 \). Let

   \[
   p_3(z) := p(z) + \frac{2Q}{\delta} h(z).
   \]

   Then \( p_3 \) is a polynomial in \( \Omega \) and \( p_3^{(k)}(\zeta_i) = p^{(k)}(\zeta_i) \) for \( i = 1, \ldots, l \) and \( k = 0, 1, \ldots, m_i \). Since \( \frac{2Q}{\delta} |h(z)| > |p(z)| \) for all \( z \in \Gamma \), Rouche’s Theorem asserts that \( p_3 \) and \( h \) have the same total number of zeros in the interior of \( \Gamma \). Therefore, \( p_3 \) has no zeros in the interior of \( \Gamma \).

This proves the theorem. \( \square \)

There exist several reliable approaches for computing \( N \). However, they all assume that some kind of *global* information is available, which may not be the case in practice and hence these algorithms are in fact of little use.
Ying and Katz developed a reliable variant of Henrici's algorithm. They assume that an upper bound for $|f''(z)|$ along an arbitrary line segment is available.

Herlocker and Ely experimented with a numerical integration approach based on Simpson's rule and the corresponding formula for the integration error. This formula involves the fourth derivative of the integrand evaluated at an unknown point in the integration interval. Automatic differentiation combined with interval arithmetic enabled them to bound the integration error.

The total number of zeros can also be computed as the topological degree of the mapping

$$F(x, y) := (\text{Re } f(x + iy), \text{Im } f(x + iy))$$

with respect to the interior of $\gamma$ (interpreted as a subset of $\mathbb{R}^2$) and the point $(0,0)$. (We will not go into the details of degree theory. For an excellent introduction, we refer the interested reader to Lloyd's book [201].) Boult and Sikorski considered the case that $\gamma$ is the boundary of the unit square $[0,1] \times [0,1]$. They proved that, in case $F$ satisfies the Lipschitz condition with constant $K > 0$ and if the infinity norm of $F$ on $\gamma$ is at least $d > 0$ where $K/(4d) \geq 1$, then at least $4[K/(4d)]$ function evaluations are needed to compute the topological degree. See also Traub, Wasilkowski and Woźniakowski [269, pp. 193–194].

As already mentioned, these reliable algorithms can seldom be used in practice since global information such as an upper bound for the modulus of a higher derivative of $f$ or the Lipschitz constant of $F = (\text{Re } f, \text{Im } f)$ or a lower bound for the infinity norm of $F$ on $\gamma$ is usually not available. Therefore, although they may indeed give incorrect results, Henrici's algorithm and numerical integration are the only approaches that we can use to compute $N$. The output of Henrici's algorithm is always an integer but unfortunately one has no idea whether it's the correct integer or not. By using quadrature formulae with different degrees of accuracy, one can easily obtain an estimate of the quadrature error. Also, the size of the imaginary part of the computed approximation for $N$ as well as the distance of the real part to the nearest integer give a clear indication of the error. (Of course, as the integral is known to be an integer, an approximation that has an error that is less than 0.5 is sufficient.) For these reasons, we prefer to use numerical integration instead of Henrici's algorithm.

1.2. An overview of other approaches. Let us conclude this section by giving a brief overview of other methods that were proposed for computing zeros of analytic functions.

Suppose that the zeros of $f$ that lie inside $\gamma$ are known to be simple. Then $f$ can be written as

$$f(z) = \phi(z) \prod_{k=1}^{n} (z - z_k)$$

(18)
where $\phi : W \to \mathbb{C}$ is analytic in $W$ and does not vanish inside $\gamma$. Let $\alpha \in \mathbb{C}$ be an arbitrary point inside $\gamma$ such that $f(\alpha) \neq 0$. Then in the interior of $\gamma$ the function $\phi$ can be written as
\begin{equation}
\phi(z) = \exp \psi(z), \quad z \in \text{int } \gamma,
\end{equation}
where $\psi$ is defined by
\begin{equation}
\psi(z) := \frac{1}{2\pi i} \int_{\gamma} \frac{\log[(t-\alpha)^{-n} f(t)]}{t-z} dt, \quad z \in \text{int } \gamma,
\end{equation}
cf. Smirnov [256]. The function $\psi$ is analytic inside $\gamma$. Note that the logarithm in (20) is well defined as the winding number of $(z-\alpha)^{-n} f(z)$ with respect to $\gamma$ and the origin is equal to zero.

Starting from (18) we find that
\begin{equation}
z_k = z - \frac{f(z)}{\phi(z) \prod_{j=1, j \neq k}^{n} (z - z_j)}
\end{equation}
for $k = 1, \ldots, n$. Assume that distinct complex numbers $\zeta_1, \ldots, \zeta_n$ are reasonably good approximations for the zeros $z_1, \ldots, z_n$ of $f$. Putting $z = \zeta_k$ and substituting the zeros $z_j$ by their approximations $\zeta_j \ (j \neq k)$ we obtain
\begin{equation}
\hat{\zeta}_k := \zeta_k - \frac{f(\zeta_k)}{\phi(\zeta_k) \prod_{j=1, j \neq k}^{n} (\zeta_k - \zeta_j)}
\end{equation}
for $k = 1, \ldots, n$. The point $\hat{\zeta}_k$ appears to be a new approximation for the zero $z_k$.

Petković, Carstensen and Trajkov [233] presented a simultaneous iterative method that is based on (21). They proved that the iteration converges quadratically if the initial approximations $\zeta_1^{(0)}, \ldots, \zeta_n^{(0)}$ are sufficiently close to the zeros $z_1, \ldots, z_n$ of $f$. The function $\phi$ is evaluated via numerical integration.

Since (18) and (19) imply that
\begin{equation}
\frac{f'(z)}{f(z)} = \psi'(z) + \sum_{k=1}^{n} \frac{1}{z - z_k}, \quad z \not\in \{z_1, \ldots, z_n\},
\end{equation}
it follows that
\begin{equation}
z_k = z - \left[ \frac{f'(z)}{f(z)} - \psi'(z) - \sum_{j=1, j \neq k}^{n} \frac{1}{z - z_j} \right]^{-1}, \quad z \not\in \{z_1, \ldots, z_n\},
\end{equation}
for $k = 1, \ldots, n$. Putting again $z = \zeta_k$ and substituting the zeros $z_j$ by their approximations $\zeta_j \ (j \neq k)$ we obtain
\begin{equation}
\hat{\zeta}_k := \zeta_k - \left[ \frac{f'(\zeta_k)}{f(\zeta_k)} - \psi'(\zeta_k) - \sum_{j=1, j \neq k}^{n} \frac{1}{\zeta_k - \zeta_j} \right]^{-1}
\end{equation}
for $k = 1, \ldots, n$. Petković and Herceg [234] analysed the corresponding iterative method and proved that it has cubic convergence. They also presented a version of
the algorithm that uses circular interval arithmetic (see also [232]). The derivative $\psi'$ is given by

$$
\psi'(z) = \frac{1}{2\pi i} \int_{\gamma} \frac{\log[(t-\alpha)^{-n} f(t)]}{(t-z)^2} dt, \quad z \in \text{int} \, \gamma,
$$

and has to be evaluated via numerical integration.

Petković and Marjanović [235] generalized these results and gave simultaneous iterative methods that have order of convergence $p + 2$ ($p = 1, 2, \ldots$) if $p$ denotes the order of the highest derivative of $f$ that appears in the iteration formula. These algorithms can be used only if the zeros of $f$ are known to be simple and if sufficiently accurate initial approximations are available. Atanassova [16] considered the case of multiple zeros but assumed that the multiplicities are known in advance.

In case $f$ is a polynomial, the iteration that corresponds to (21) is called the Durand-Kerner (or also Weierstrass) method whereas the iteration that corresponds to (22) is known as Aberth's method (cf. Bini [32]). The problem of calculating zeros of polynomials received a lot of interest in the past and is still a lively area of research. McNamee [208, 209] compiled an extensive bibliography on computing zeros of polynomials. Although polynomials are of course a special case of analytic functions, we will not give a comprehensive overview of all the methods that were proposed. Instead we refer the interested reader to the papers by Bini and Pan [36], Cardinal [52], Carstensen and Sakurai [55] and the survey paper by Pan [229] as well as the references cited therein. Recently, Bini, Gemignani and Meini [33] have presented an interesting method to compute a factor of a polynomial or of an analytic function that is given as a power series. Their approach is based on a matrix version of Koenig's Theorem and on cyclic reduction. It involves infinite Toeplitz matrices in block Hessenberg form.

In a number of papers and short notes, Anastasselou and Ioakimidis [7, 8, 9, 10, 11, 159, 160, 162, 163] considered the problem of computing zeros of sectionally analytic functions (i.e., functions that are analytic except for a finite number of discontinuity arcs). They proposed variations and generalizations of the method of Burniston and Siewert [48], which is based on the theory of Riemann-Hilbert boundary value problems (cf. Gakhov [87]). The authors focused on the function $\alpha + \beta z - z \tanh^{-1}(1/z)$ where $\alpha, \beta \in \mathbb{C}$ are parameters. This function appears in the theory of ferromagnetism. It has the discontinuity interval $[-1, 1]$. Already for this example, the approach of Anastasselou and Ioakimidis requires a lot of specific analytical calculations. Therefore, it is unclear how their method could lead to a 'black box' algorithm that can handle arbitrary functions. Also, although multiple zeros are not a problem, their approach cannot calculate multiplicities.

Yakoubsohn [305] proposed an exclusion method for computing zeros of analytic functions. Unfortunately, his exclusion function is difficult to evaluate and multiple zeros require special treatment. He applied his algorithm only to polynomials. See also Ying and Katz [308].
2. Formal orthogonal polynomials

Let $\mathcal{P}$ be the linear space of polynomials with complex coefficients. We define a symmetric bilinear form

$$\langle \cdot, \cdot \rangle : \mathcal{P} \times \mathcal{P} \to \mathbb{C}$$

by setting

$$\langle \phi, \psi \rangle = \frac{1}{2\pi i} \int_\gamma \phi(z)\psi(z)\frac{f'(z)}{f(z)} \, dz = \sum_{k=1}^{n} \nu_k \phi(z_k)\psi(z_k)$$

for any two polynomials $\phi, \psi \in \mathcal{P}$. The latter equality follows from the fact that $f'/f$ has a simple pole at $z_k$ with residue $\nu_k$ for $k = 1, \ldots, n$. Note that $\langle \cdot, \cdot \rangle$ can be evaluated via numerical integration along $\gamma$. In what follows, we will assume that all the “inner products” $\langle \phi, \psi \rangle$ that are needed have been calculated. Let $s_p := (1, z^p)$ for $p = 0, 1, 2, \ldots$. These ordinary moments are equal to the Newton sums of the unknown zeros,

$$s_p = \sum_{k=1}^{n} \nu_k z_k^p, \quad p = 0, 1, 2, \ldots$$

In particular, $s_0 = \nu_1 + \cdots + \nu_n = N$, the total number of zeros. Hence, we may assume that the value of $N$ is known. Let $H_k$ be the $k \times k$ Hankel matrix

$$H_k := \left[ \begin{array}{cccc} s_0 & s_1 & \cdots & s_{k-1} \\ s_1 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ s_{k-1} & \cdots & \cdots & s_{2k-2} \end{array} \right]$$

for $k = 1, 2, \ldots$. Let $H$ be the infinite Hankel matrix

$$H := [s_{p+q}]_{p, q \geq 0}.$$ 

Observe that the form $\langle \cdot, \cdot \rangle$ is completely determined by the sequence of moments $(s_p)_{p \geq 0}$.

A monic polynomial $\varphi_t$ of degree $t \geq 0$ that satisfies

$$\langle z^k, \varphi_t(z) \rangle = 0, \quad k = 0, 1, \ldots, t-1,$$

is called a formal orthogonal polynomial (FOP) of degree $t$. (Observe that condition (24) is void for $t = 0$.) The adjective formal emphasizes the fact that, in general, the form $\langle \cdot, \cdot \rangle$ does not define a true inner product. An important consequence of this fact is that, in contrast to polynomials that are orthogonal with respect to a true inner product, formal orthogonal polynomials need not exist or need not be unique for every degree. (For details, see for example Draux [71, 72], Gutknecht [128, 130] or Garg and Gutknecht [113].) If (24) is satisfied and $\varphi_t$ is unique, then $\varphi_t$ is called a regular FOP and $t$ a regular index. If we set

$$\varphi_t(z) = u_{0,t} + u_{1,t}z + \cdots + u_{t-1,t}z^{t-1} + z^t$$
then condition (24) translates into the Yule-Walker system

\[
\begin{bmatrix}
    s_0 & s_1 & \cdots & s_{t-1} \\
    s_1 & \ddots & \vdots & \vdots \\
    \vdots & \ddots & \vdots & \vdots \\
    s_{t-1} & \cdots & s_{2t-2} & s_{2t-1}
\end{bmatrix}
\begin{bmatrix}
    u_{0,t} \\
    u_{1,t} \\
    \vdots \\
    u_{t-1,t}
\end{bmatrix}
= -
\begin{bmatrix}
    s_t \\
    s_{t+1} \\
    \vdots \\
    s_{2t-1}
\end{bmatrix}
\]

(25)

Hence, the regular FOP of degree \( t \geq 1 \) exists if and only if the matrix \( H_t \) is nonsingular. Thus, the rank profile of \( H \) determines which regular FOPs exist. If \( t \) is a regular index, then

\[
\varphi_t(z) = \frac{1}{\det H_t} \begin{vmatrix}
    s_0 & s_1 & \cdots & s_{t-1} & 1 \\
    s_1 & \ddots & \vdots & \vdots & \vdots \\
    \vdots & \ddots & \ddots & \vdots & \vdots \\
    s_{t-1} & \cdots & s_{2t-2} & s_{2t-1} & z^{t-1} \\
    s_t & \cdots & s_{2t-1} & z^{t-1} & \vdots
\end{vmatrix},
\]

as one can easily verify. Note that this determinant expression implies that

\[
\langle \varphi_t, \varphi_t \rangle = \frac{\det H_{t+1}}{\det H_t}.
\]

The following theorem characterizes \( n \), the number of mutually distinct zeros. It enables us, theoretically at least, to calculate \( n \) as \( \text{rank} H_N \).

**Theorem 3.** \( n = \text{rank} H_{n+p} \) for every nonnegative integer \( p \). In particular, \( n = \text{rank} H_N \).

**Proof.** Let \( p \) be a nonnegative integer. The matrix \( H_{n+p} \) can be written as

\[
H_{n+p} = \sum_{k=1}^{n} \nu_k \begin{bmatrix}
    1 & \cdots & z_k^{n+p-1} \\
    \vdots & \ddots & \vdots \\
    z_k^{n+p-1} & \cdots & z_k^{2(n+p)-2}
\end{bmatrix}
\]

This implies that \( \text{rank} H_{n+p} \leq n \). However, \( H_n \) is nonsingular. Indeed, as one can easily verify, the matrix \( H_n \) can be factorized as \( H_n = V_n D_n V_n^T \) where \( V_n \) is the Vandermonde matrix \( V_n := [z_k]_{r=0, s=1}^{n} \) and \( D_n \) is the diagonal matrix \( D_n := \text{diag}(\nu_1, \ldots, \nu_n) \). Therefore \( \text{rank} H_{n+p} \geq n \). It follows that \( \text{rank} H_{n+p} = n \).

Therefore \( H_n \) is nonsingular whereas \( H_t \) is singular for \( t > n \). Note that \( H_1 = [s_0] \) is nonsingular by assumption. The regular FOP of degree 1 exists and is given by \( \varphi_1(z) = z - \mu \) where

\[
\mu := \frac{s_1}{s_0} = \frac{\sum_{k=1}^{n} \nu_k z_k}{\sum_{k=1}^{n} \nu_k}
\]
is the arithmetic mean of the zeros. Theorem 3 implies that the regular FOP \( \varphi_n \) of degree \( n \) exists and tells us also that regular FOPs of degree larger than \( n \) do not exist. The polynomial \( \varphi_n \) is easily seen to be

\[
(27) \quad \varphi_n(z) = (z - z_1) \cdots (z - z_n).
\]

It is the monic polynomial of degree \( n \) that has \( z_1, \ldots, z_n \) as simple zeros. Its coefficients can be calculated by solving an \( n \times n \) Yule-Walker system. This polynomial has the peculiar property that it is orthogonal to all polynomials (including itself),

\[
(28) \quad \langle z^k, \varphi_n(z) \rangle = 0, \quad k = 0, 1, 2, \ldots
\]

**Note.** Kronecker’s Theorem [230, p.37] tells us that the infinite Hankel matrix \( H \) has finite rank if and only if its *symbol*, which is defined as the formal Laurent series

\[
\frac{s_0}{z} + \frac{s_1}{z^2} + \frac{s_2}{z^3} + \cdots,
\]

represents a rational function of \( z \). This is indeed the case. It is easily seen that

\[
(29) \quad \sum_{k=1}^{n} \frac{\nu_k}{z - z_k} = \frac{s_0}{z} + \frac{s_1}{z^2} + \frac{s_2}{z^3} + \cdots \quad \text{near } z = \infty.
\]

In system theory the problem of reconstructing the rational function that appears in the left-hand side of (29) from the sequence of moments \( (s_p)_{p \geq 0} \) is called a minimal realization problem. In that context the moments are called Markov parameters. For more details and the connection with continued fractions, Padé approximation, the Euclidean algorithm for formal Laurent series and the Berlekamp-Massey algorithm, see for example [45, 46, 47, 71, 113, 166]. Note that the left-hand side of (29) is a rational function of type \([n - 1/n]\) and that its denominator polynomial is given by \( \varphi_n(z) \).

Once \( n \) is known, the mutually distinct zeros \( z_1, \ldots, z_n \) can be calculated by solving a generalized eigenvalue problem. Indeed, let \( H_n^< \) be the Hankel matrix

\[
H_n^< := \begin{bmatrix}
  s_1 & s_2 & \cdots & s_n \\
  s_2 & \ddots & \vdots & \\
  \vdots & \ddots & \ddots & \vdots \\
  s_n & \cdots & s_{n-1} & s_{2n-1}
\end{bmatrix}
\]

**Theorem 4.** The eigenvalues of the pencil \( H_n^< - \lambda H_n \) are given by \( z_1, \ldots, z_n \).

**Proof.** Suppose that \( \varphi_n(z) = u_0 n + u_{1,n} z + \cdots + u_{n-1,n} z^{n-1} + z^n \). Then (27) implies that the zeros \( z_1, \ldots, z_n \) are given by the eigenvalues of the companion matrix

\[
C_n := \begin{bmatrix}
  0 & 0 & \cdots & 0 & -u_{0,n} \\
  1 & 0 & \cdots & 0 & -u_{1,n} \\
  \vdots & \ddots & \vdots & \vdots & \vdots \\
  \vdots & \ddots & 0 & -u_{n-2,n} \\
  0 & \cdots & 0 & 1 & -u_{n-1,n}
\end{bmatrix}.
\]
Let $\lambda^*$ be an eigenvalue of $C_n$ and $x$ a corresponding eigenvector. As $H_n$ is regular, we may conclude that

$$C_nx = \lambda^* x \iff H_nC_nx = \lambda^* H_nx.$$  

Using (25) one can easily verify that $H_nC_n = H_n^\prec$. This proves the theorem.

Another proof goes as follows. As in the proof of Theorem 3, let $V_n$ be the Vandermonde matrix

$$V_n := \begin{bmatrix} 1 & \cdots & 1 \\ z_1 & \cdots & z_n \\ \vdots & \ddots & \vdots \\ z_1^{n-1} & \cdots & z_n^{n-1} \end{bmatrix}$$

and let $D_n := \text{diag}(\nu_1, \ldots, \nu_n)$. Also, let $D_n^{(1)} := \text{diag}(\nu_1 z_1, \ldots, \nu_n z_n)$. Then the matrices $H_n$ and $H_n^\prec$ can be factorized as

$$H_n = V_n D_n V_n^T \quad \text{and} \quad H_n^\prec = V_n D_n^{(1)} V_n^T.$$  

Let $\lambda^*$ be an eigenvalue of the pencil $H_n^\prec - \lambda H_n$ and $x$ a corresponding eigenvector. Then

$$\begin{align*} H_n^\prec x &= \lambda^* H_n x \\ \iff & V_n D_n^{(1)} V_n^T x = \lambda^* V_n D_n V_n^T x \\ \iff & D_n^{(1)} y = \lambda^* D_n y \quad \text{if } y := V_n^T x \\ \iff & \text{diag}(z_1, \ldots, z_n) y = \lambda^* y. \end{align*}$$

This proves the theorem. $\square$

Note. Recently, Golub, Milanfar and Varah [111] have presented a stable numerical solution to the problem of reconstructing a polygonal shape from moments. This problem has many applications including tomographic reconstruction and geophysical inversion. In the latter application, it is of interest to reconstruct the shape and (possibly) density of a gravitational anomaly from discrete measurements of the exterior gravitational field at spatially separated points. The authors’ approach is based on the matrix pencil that appears in Theorem 4. To solve the generalized eigenvalue problem, they have developed an efficient variant of the QZ algorithm that takes into account the way the matrices $H_n$ and $H_n^\prec$ are related.

Once $z_1, \ldots, z_n$ have been found, the multiplicities $\nu_1, \ldots, \nu_n$ can be computed by solving the Vandermonde system

$$\begin{bmatrix} 1 & \cdots & 1 \\ z_1 & \cdots & z_n \\ \vdots & \ddots & \vdots \\ z_1^{n-1} & \cdots & z_n^{n-1} \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \\ \vdots \\ \nu_n \end{bmatrix} = \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{n-1} \end{bmatrix}. \tag{30}$$

This can be done via the algorithm of Gohberg and Koltracht [108]. This algorithm takes full account of the structure of a Vandermonde matrix and is not only faster but also more accurate than general purpose algorithms such as Gaussian elimination with partial pivoting. It has arithmetic complexity $O(n^3)$. 
Vandermonde matrices are often very ill-conditioned. Gautschi wrote many papers on this subject, see for example [61, 91, 92, 94, 95, 98, 99]. In our case, however, the components of the solution vector of (30) are known to be integers, and therefore there is no problem, even if the linear system (30) happens to be ill-conditioned, as long as the computed approximations for the components of the solution vector have an absolute error that is less than 0.5. We remark that Hankel matrices also have the reputation of being ill-conditioned [271].

Theorem 3 and 4 suggest the following approach to compute $n$ and $z_1, \ldots, z_n$. Start by computing the total number of zeros $N$. Next, compute $s_1, \ldots, s_{2N-2}$. As already mentioned, this can be done via numerical integration along $\gamma$. The number of mutually distinct zeros is then calculated as the rank of $H_N$, $n = \text{rank} \, H_N$. Finally, the zeros $z_1, \ldots, z_n$ are obtained by solving a generalized eigenvalue problem.

Unfortunately, this approach has several disadvantages:

1. Theoretically the $N - n$ smallest singular values of $H_N$ are equal to zero. In practice, this will not be the case, and it may be difficult to determine the rank of $H_N$ and hence the value of $n$ in case the gap between the computed approximations for the zero singular values and the nonzero singular values is too small.
2. The approximations for $z_1, \ldots, z_n$ obtained via Theorem 4 may not be very accurate. Indeed, the mapping from the Newton sums to the zeros and their respective multiplicities,

\[(s_0, s_1, \ldots, s_{2n-1}) \mapsto (z_1, \ldots, z_n, \nu_1, \ldots, \nu_n),\]

is usually very ill-conditioned. (See for example the papers by Gautschi [93, 96, 97] who studied the conditioning of (31) in the context of Gauss quadrature formulae. For a recent paper on this subject, we refer to Beckermann and Bourreau [19].) Indeed, a classical adage in numerical analysis says that one should avoid the use of ordinary moments.

In Section 3 we will present an algorithm that gives more accurate approximations for $z_1, \ldots, z_n$. The idea is the following. The inner products that appear in the Hankel matrices $H_n$ and $H_n^< \leq$ are related to the standard monomial basis. Why not consider a different basis? In other words, let us try to use modified moments instead of ordinary moments. The fact that

\[H_n = \left[\langle z^p, z^q \rangle \right]_{p,q=0}^{n-1} \quad \text{and} \quad H_n^< \leq = \left[\langle z^p, z^q \rangle \right]_{p,q=0}^{n-1} \]

suggests that we should consider the matrices

\[(\psi_0, \psi_1, \ldots, \psi_{2n-1}^< \leq)_{p,q=0}^{n-1} \quad \text{and} \quad (\psi_0^< \leq, \psi_1^< \leq, \ldots, \psi_n^< \leq)_{p,q=0}^{n-1} \]

where $\psi_k$ is a polynomial of degree $k$ for $k = 0, 1, \ldots, n - 1$. Of course, even if we succeed in writing Theorem 4 in terms of (32), the question remains which polynomials $\psi_k$ to choose. We have found that very accurate results are obtained if we use the formal orthogonal polynomials. In other words, the zeros of $\varphi_n(z)$ will be computed from inner products that involve $\varphi_0(z), \varphi_1(z), \ldots, \varphi_{n-1}(z)$. The value of $n$ will be determined indirectly.
All this will be explained in more detail in Section 3. Let us conclude this section by discussing the orthogonality properties of FOPs. This will enable us to define the matrices $\{\varphi_0, \varphi_1, \ldots, \varphi_n\}$ of regular FOPs.

If $H_n$ is strongly nonsingular, i.e., if all its leading principal submatrices are nonsingular, then we have a full set $\{\varphi_0, \varphi_1, \ldots, \varphi_n\}$ of regular FOPs.

What happens if $H_n$ is not strongly nonsingular and thus there is no full set of regular FOPs? Let $\{k_j\}_{j=0}^t$ be the set of all regular indices, with

$$k_0 < k_1 < \cdots < k_J.$$

Then $k_0 = 0$, $k_1 = 1$ and $k_J = n$. By filling up the gaps in the sequence of existing regular FOPs it is possible to define a sequence $\{\varphi_t\}_{t=0}^\infty$, with $\varphi_t$ a monic polynomial of degree $t$, such that if these polynomials are grouped into blocks according to the sequence of regular indices, then polynomials belonging to different blocks are orthogonal with respect to (23). More precisely, define $\{\varphi_t\}_{t=0}^\infty$ as follows. If $t$ is a regular index, then let $\varphi_t$ be the regular FOP of degree $t$. Else define $\varphi_t$ as $\varphi_r \psi_{t,r}$, where $r$ is the largest regular index less than $t$ and $\psi_{t,r}$ is an arbitrary monic polynomial of degree $t-r$. In the latter case $\varphi_t$ is called an inner polynomial. If $\psi_{t,r}(z) = z^{t-r}$ then we say that $\varphi_t$ is defined by using the standard monomial basis.

These polynomials $\{\varphi_t\}_{t=0}^\infty$ can be grouped into $J+1$ blocks

$$\Phi^{(0)} := \begin{bmatrix} \varphi_0 \end{bmatrix}, \quad \Phi^{(1)} := \begin{bmatrix} \varphi_1 & \varphi_2 & \cdots & \varphi_{k_2-1} \end{bmatrix}, \quad \Phi^{(2)} := \begin{bmatrix} \varphi_{k_2} & \varphi_{k_2+1} & \cdots & \varphi_{k_3-1} \end{bmatrix}, \quad \vdots \quad \vdots \quad \ddots \quad \vdots \quad \Phi^{(J-1)} := \begin{bmatrix} \varphi_{k_{J-1}} & \varphi_{k_{J-1}+1} & \cdots & \varphi_{k_J-1} \end{bmatrix}, \quad \Phi^{(J)} := \begin{bmatrix} \varphi_n & \varphi_{n+1} & \cdots & \end{bmatrix}.$$  

Note that each block starts with a regular FOP and that the remaining polynomials in each block are inner polynomials. The $p$th block has length $l_p := k_{p+1} - k_p$ for $p = 0, 1, \ldots, J-1$. Let

$$\langle \Psi, \Phi \rangle := \begin{bmatrix} \langle \psi_0, \phi_0 \rangle & \cdots & \langle \psi_0, \phi_q \rangle \\ \vdots & \ddots & \vdots \\ \langle \psi_p, \phi_0 \rangle & \cdots & \langle \psi_p, \phi_q \rangle \end{bmatrix} \in \mathbb{C}^{(p+1) \times (q+1)}$$

for any two row vectors

$$\Psi := \begin{bmatrix} \psi_0 & \psi_1 & \cdots & \psi_p \end{bmatrix} \quad \text{and} \quad \Phi := \begin{bmatrix} \phi_0 & \phi_1 & \cdots & \phi_q \end{bmatrix}$$

of polynomials in $\mathcal{P}$.

**Theorem 5.** The following block orthogonality relations hold:

$$\langle \Phi^{(p)}, \Phi^{(q)} \rangle = \begin{cases} 0_{l_p \times l_q} & \text{if } p \neq q \\ \delta_p & \text{if } p = q \end{cases}$$

for $p, q = 0, 1, \ldots, J-1$. The matrix $\delta_p \in \mathbb{C}^{l_p \times l_p}$ is nonsingular and symmetric for $p = 0, 1, \ldots, J-1$. Its entries are equal to zero above the main antidiagonal and equal to $\langle z^{k_p + l_p - 1}, \varphi_{l_p} \rangle$.
along the main antidiagonal. Also, if all the inner polynomials of the block \( \Phi^{(p)} \) where \( p \in \{1, \ldots, J-1\} \) are defined by using the standard monomial basis, then \( \delta_p \) is a Hankel matrix.

**Proof.** This result is well-known in the theory of FOPs. However, as readers who are less familiar with formal orthogonal polynomials than with the literature concerning the computation of zeros may find themselves slightly overwhelmed by the amount of information given in, for example, the survey papers by Gutknecht [128, 130] or the book by Bultheel and Van Barel [47], we prefer to include a proof.

The proof is by induction. Obviously, \( \langle \Phi^{(0)}, \Phi^{(0)} \rangle = [(1,1)] = [s_0] \) is nonsingular. Now suppose that the theorem holds for \( p, q = 0, 1, \ldots, k-1 \) where \( k \in \{1, \ldots, J-1\} \). Consider the block \( \Phi^{(k)} \). Let us call the first polynomial of this block \( \varphi_r \) and let \( l \) be the length of this block,

\[
\Phi^{(k)} = \begin{bmatrix}
\varphi_r & \varphi_{r+1} & \cdots & \varphi_{r+l-1}
\end{bmatrix}.
\]

Then the matrices \( H_{r+1}, \ldots, H_{r+l-1} \) are singular while \( H_r \) and \( H_{r+l} \) are nonsingular. By symmetry considerations it suffices to prove that

\[
K := \begin{bmatrix}
\langle \varphi_0, \varphi_r \rangle & \langle \varphi_0, \varphi_{r+l-1} \rangle \\
\langle \varphi_1, \varphi_r \rangle & \langle \varphi_1, \varphi_{r+l-1} \rangle \\
\vdots & \vdots \\
\langle \varphi_{r-1}, \varphi_r \rangle & \langle \varphi_{r-1}, \varphi_{r+l-1} \rangle \\
\end{bmatrix} = 0_{r \times l}
\]

and that the matrix

\[
\delta := \begin{bmatrix}
\langle \varphi_r, \varphi_r \rangle & \langle \varphi_r, \varphi_{r+l-1} \rangle \\
\vdots & \vdots \\
\langle \varphi_{r+l-1}, \varphi_r \rangle & \langle \varphi_{r+l-1}, \varphi_{r+l-1} \rangle \\
\end{bmatrix}
\]

is nonsingular and has the other properties mentioned. Let \( F_k \) be the \( l \times l \) unit upper triangular matrix that contains the coefficients of the polynomials

\[
1, \varphi_{r+1}, \ldots, \varphi_{r+l-1, r}
\]

(used in the definition of the inner polynomials of the block \( \Phi^{(k)} \)). Then

\[
K = K F_k \quad \text{and} \quad \delta = F_k^T \delta F_k
\]

if the matrices \( K \) and \( \delta \) are defined as

\[
K := \begin{bmatrix}
\langle \varphi_0, \varphi_r \rangle & \langle \varphi_0, \varphi_{r+l-1} \rangle \\
\langle \varphi_1, \varphi_r \rangle & \langle \varphi_1, \varphi_{r+l-1} \rangle \\
\vdots & \vdots \\
\langle \varphi_{r-1}, \varphi_r \rangle & \langle \varphi_{r-1}, \varphi_{r+l-1} \rangle \\
\end{bmatrix}
\]

and

\[
\delta := \begin{bmatrix}
\langle \varphi_r, \varphi_r \rangle & \langle \varphi_r, \varphi_{r+l-1} \rangle \\
\langle \varphi_r, \varphi_r \rangle & \langle \varphi_r, \varphi_{r+l-1} \rangle \\
\vdots & \vdots \\
\langle \varphi_{r+l-1}, \varphi_r \rangle & \langle \varphi_{r+l-1}, \varphi_{r+l-1} \rangle \\
\end{bmatrix}
\]

(In other words, \( K \) and \( \delta \) correspond to the situation where all the inner polynomials in our block are defined by using the standard monomial basis.) Therefore we will
first study the matrices $K$ and $\delta$. Observe that $\delta$ is a Hankel matrix. As $\varphi_r$ is a regular FOP, we may conclude that $\langle \varphi_s, \varphi_t \rangle = 0$ for $t \geq 0$ and $s = 0, 1, \ldots, r - t - 1$. Thus

$$K = \begin{bmatrix}
0 & \cdots & 0 \\
\vdots & \ddots & \vdots \\
\vdots & & \ddots & \vdots \\
0 & \cdots & 0 \\
\vdots & & \cdots & \ddots \\
0 & \cdots & \cdots & \cdots & \ddots \\
\end{bmatrix}.$$  

(33)

Let us consider the first antidiagonal of $K$ whose entries we have not yet proven to be equal to zero. As $\varphi_r$ is orthogonal to all polynomials of degree $\leq r - 1$, all these entries are equal. Indeed,

$$\langle \varphi_{r-1}, z \varphi_r \rangle = \langle \varphi_{r-2}, z^2 \varphi_r \rangle = \cdots = \langle \varphi_{r-l+1}, z^{l-1} \varphi_r \rangle = \langle z^r, \varphi_r \rangle.$$  

Note that $\langle z^r, \varphi_r \rangle = \langle \varphi_r, \varphi_r \rangle$, the entry in the upper left corner of $\delta$. As $H_{r+1}$ is singular and $\langle \varphi_r, \varphi_r \rangle \neq 0$, it follows that $\det H_r = 0$. This implies that $\varphi_r$ is in fact orthogonal to all polynomials of degree $\leq r$ and that the upper left entry of $\delta$ as well as all the entries on our antidiagonal of $K$ are equal to zero. Now we continue with the next antidiagonal of $K$. The fact that $\langle \varphi_r, \varphi_r \rangle = 0$ implies that all its entries are equal to $\langle z^{r+1}, \varphi_r \rangle = \langle z \varphi_r, \varphi_r \rangle$. One can easily check that

$$R_{r+2}^T H_{r+2} R_{r+2} = \left[ \langle \varphi_r, \varphi_r \rangle \right]_{p,q=0}^{r+1}$$

$$= \text{diag}(\delta_0, \delta_1, \ldots, \delta_{l-1}) \oplus F_2^T \begin{bmatrix} 0 & (z^{r+1}, \varphi_r) \\ (z^{r+1}, \varphi_r) & (z \varphi_r, \varphi_r) \end{bmatrix} F_2$$

if $R_{r+2}$ is the unit upper triangular matrix that contains the coefficients of the polynomials $\varphi_0, \varphi_1, \ldots, \varphi_r$. As $H_{r+2}$ is singular and $\delta_0, \delta_1, \ldots, \delta_{l-1}$ are nonsingular, it follows that $\langle z^{r+1}, \varphi_r \rangle = 0$. Thus $\varphi_r$ is orthogonal to all polynomials of degree $\leq r + 1$ and all the entries on our antidiagonal of $K$ as well as two additional entries of $\delta$ are equal to zero. And so on. Eventually we will find that all the entries of $K$ that are marked $\times$ in (33) are determined by the first $l - 1$ entries of the first row of $\delta$ “in a Hankel way,” i.e., by shifting these entries to the north-east. We will also find that $\langle z^{r+2}, \varphi_r \rangle = \cdots = \langle z^{r+l-2}, \varphi_r \rangle = 0$, i.e., $\varphi_r$ is orthogonal to all polynomials of degree $\leq r + l - 2$ (because $H_{r+3}, \ldots, H_{r+l-1}$ are singular) while $\langle z^{r+l-1}, \varphi_r \rangle \neq 0$ (because $H_{r+l-1}$ is nonsingular). Therefore $K = 0_{r \times r}$ and $\delta$ is a nonsingular lower triangular Hankel matrix. One can easily verify that this implies that $\delta$ is indeed nonsingular, symmetric, zero above the main antidiagonal, and equal to $\langle z^{r+l-1}, \varphi_r \rangle$ along the main antidiagonal. This proves the theorem.  

□

Note that $\langle \Phi^{(p)}, \Phi^{(J)} \rangle = 0_{p \times \infty}$ for $p = 0, 1, \ldots, J$ if we set $l_J := \infty$.

Let $G$ be the infinite Gram matrix

$$G := \left[ \langle \varphi_p, \varphi_q \rangle \right]_{p,q \geq 0}$$
and let $G_k$ be its $k \times k$ leading principal submatrix for $k = 1, 2, \ldots$. Then
\[
G = G_n \oplus 0_{\infty \times \infty}.
\]

The matrix $G_n$ is nonsingular and block diagonal, $G_n = \text{diag}(\delta_0, \delta_1, \ldots, \delta_{J-1})$. Let $G^{(1)}$ be the infinite matrix
\[
G^{(1)} := \{(\varphi_p, \varphi_q)\}_{p,q \geq 0}
\]
and let $G^{(1)}_k$ be its $k \times k$ leading principal submatrix for $k = 1, 2, \ldots$. Then
\[
G^{(1)} = G^{(1)}_n \oplus 0_{\infty \times \infty}.
\]

Note that $G$ and $G^{(1)}$ are both symmetric. As already mentioned, the matrices $G_n$ and $G^{(1)}_n$ will play an important role in Section 3. The following theorem examines the structure of $G^{(1)}_n$. Let us agree to call a matrix $A = [a_{p,q}]_{p,q=0}^{\infty} \in \mathbb{C}^{\infty \times \infty}$ lower anti-Hessenberg if $a_{p,q} = 0$ whenever $p + q < l - 2$, i.e., a matrix will be called lower anti-Hessenberg if its entries are equal to zero along all the antidiagonals that lie above the main antidiagonal, except for the antidiagonal that precedes the main antidiagonal.

**Theorem 6.** The following block orthogonality relations hold:
\[
\langle \Phi^{(p)}, \varphi_1 \Phi^{(q)} \rangle = \begin{cases}
0_{p \times q} & \text{if } |p - q| > 1 \\
\kappa_p & \text{if } p = q + 1 \\
\kappa_q^T & \text{if } p = q - 1 \\
\delta_p^{(1)} & \text{if } p = q
\end{cases} \quad \text{for } p, q = 0, 1, \ldots, J - 1.
\]

In other words, the matrix $G^{(1)}_n$ is block tridiagonal,
\[
G^{(1)}_n = \begin{bmatrix}
\delta^{(1)}_0 & \kappa_1^T & & \\
\kappa_1 & \delta^{(1)}_1 & \kappa_2^T & \\
& \ddots & \ddots & \\
& & \kappa_{J-2} & \delta^{(1)}_{J-2} & \kappa_{J-1}^T \\
& & & \kappa_{J-2} & \delta^{(1)}_{J-1} \\
\end{bmatrix}
\]

The matrix $\delta_p^{(1)} \in \mathbb{C}_p^{\infty \times \infty}$ is symmetric and lower anti-Hessenberg for $p = 0, 1, \ldots, J - 1$. Its entries are equal to $\langle z^{p+q-1}, \varphi_p \rangle$ along its first nonzero antidiagonal. Also, if all the inner polynomials of the block $\Phi^{(p)}$ where $p \in \{1, \ldots, J - 1\}$ are defined by using the standard monomial basis, then $\delta^{(1)}_p$ is a Hankel matrix. Note that $\delta^{(1)}_0 = [0]$. All the entries of the matrix $\kappa_p$ where $p \in \{1, \ldots, J - 1\}$ are equal to zero, except for the entry in the south-east corner, which is equal to $\langle z^{p+q-1}, \varphi_p \rangle$.

**Proof.** The proof is left to the reader. Use the results obtained in the proof of Theorem 5. \qed
Thus, for example, the matrices $G_n$ and $G_n^{(1)}$ may look like

$$G_n = \begin{bmatrix}
\odot \\
0 & 0 & \odot \\
0 & \odot & \times \\
\odot & \times & \times \\
0 & 0 & \odot \\
0 & \odot & \times \\
0 & \odot & \times \\
\odot & \times & \times \\
\end{bmatrix}$$

and

$$G_n^{(1)} = \begin{bmatrix}
0 & \odot \\
0 & \odot & \times \\
\odot & \times & \times \\
\odot & \times & \times \\
0 & \odot & \times \\
0 & \odot & \times \\
0 & \odot & \times \\
\odot & \times & \times \\
\end{bmatrix}.$$
Theorem 7. Let $t \geq 1$ be a regular index and let $z_{1,1}, \ldots, z_{t,t}$ be the zeros of the regular FOP $\varphi_t$. Then the eigenvalues of the pencil $G_t^{(1)} - \lambda G_t$ are given by $\varphi_1(z_{1,1}), \ldots, \varphi_1(z_{t,t})$. In other words, they are given by $z_{1,1} - \mu, \ldots, z_{t,t} - \mu$ where $\mu = s_1/s_0$.

Proof. The first part of the proof is similar to the proof of Theorem 4. Define the Hankel matrix $H_t^\prec$ as $H_t^\prec := [s_{1+k+l}]_{k,l=0}^{t-1}$. We will first show that the zeros of $\varphi_t$ are given by the eigenvalues of the pencil $H_t^\prec - \lambda H_t$. The zeros of $\varphi_t$ are given by the eigenvalues of its companion matrix $C_t$. Let $\lambda^*$ be an eigenvalue of $C_t$ and $x$ a corresponding eigenvector. As $H_t$ is nonsingular, we may conclude that $C_t x = \lambda^* x \iff H_t C_t x = \lambda^* H_t x$. Using (25) one can easily verify that $H_t C_t = H_t^\prec$.

Let $A_t$ be the unit upper triangular matrix that contains the coefficients of $\varphi_0, \varphi_1, \ldots, \varphi_{t-1}$. Then $G_t$ can be factorized as $G_t = A_t^T H_t A_t$. As $\varphi_1(z) = z - \mu$ where $\mu = s_1/s_0$, the matrix $G_t^{(1)}$ is given by $[(\varphi_r, z \varphi_l)]_{r,s=0}^{t-1} = -\mu G_t$. The matrix $[(\varphi_r, z \varphi_l)]_{r,s=0}^{t-1} = \mu$ can be written as $A_t^T H_t^\prec A_t$ and thus $G_t^{(1)} = A_t^T (H_t^\prec - \mu H_t) A_t$. Now let $\lambda^*$ be an eigenvalue of the pencil $H_t^\prec - \lambda H_t$ and $x$ a corresponding eigenvector. Then

$$
H_t^\prec x = \lambda^* H_t x \\
\iff (H_t^\prec - \mu H_t)x = (\lambda^* - \mu)H_t x \\
\iff A_t^T (H_t^\prec - \mu H_t) A_t y = \varphi_1(\lambda^*) A_t^T H_t A_t y \quad \text{if } y := A_t^{-1} x \\
\iff G_t^{(1)} y = \varphi_1(\lambda^*) G_t y.
$$

This proves the theorem. \qed

Corollary 8. The eigenvalues of $G_n^{(1)} - \lambda G_n$ are given by $z_1 - \mu, \ldots, z_n - \mu$ where $\mu = s_1/s_0$.

Regular FOPs are characterized by the fact that the determinant of a Hankel matrix is different from zero, while inner polynomials correspond to singular Hankel matrices. To decide whether $\varphi_t(z)$ should be defined as a regular FOP or as an inner polynomial, one could therefore calculate the determinant of $H_t$ and check if it is equal to zero. However, from a numerical point of view such a test "is equal to zero" does not make sense. Because of rounding errors (both in the evaluation of $\langle \cdot, \cdot \rangle$ and the calculation of the determinant) we would encounter only regular FOPs. Strictly speaking one could say that inner polynomials are not needed in numerical calculations. However, the opposite is true! Let us agree to call a regular FOP well-conditioned if its corresponding Yule-Walker system (25) is well-conditioned, and ill-conditioned otherwise. To obtain a numerically stable algorithm, it is crucial to generate only well-conditioned regular FOPs and to replace ill-conditioned regular FOPs by inner polynomials. Stable look-ahead solvers for linear systems of equations that have Hankel structure are based on this principle [39, 51, 86]. In this approach the diagonal blocks in $G_n$ are taken (slightly) larger than strictly necessary to avoid ill-conditioned blocks. A disadvantage is that part of the structure of $G_n$ and $G_n^{(1)}$ gets lost, i.e., there will be some additional fill-in.

Our algorithm for calculating $z_1, \ldots, z_n$ proceeds by computing the polynomials $\varphi_0(z), \varphi_1(z), \ldots, \varphi_n(z)$ in their product representation, starting with $\varphi_0(z) \leftarrow 1$ and
\( \varphi_1(z) \leftarrow z - \mu \). At each step we ask ourselves whether it is numerically feasible to generate the next polynomial in the sequence as a regular FOP. As the reader will see, there are several ways to find an answer to this question.

How do we obtain the value of \( n \)? Theorem 3 and Equations (23) and (27) imply the following.

**Theorem 9.** Let \( t \geq n \). Then \( \varphi_t(z_k) = 0 \) for \( k = 1, \ldots, n \) and \( \langle z^p, \varphi_t(z) \rangle = 0 \) for all \( p \geq 0 \).

The value of \( n \) can be determined as follows. Suppose that the algorithm has just generated a (well-conditioned) regular FOP \( \varphi_r(z) \). To check whether \( n = r \), we scan the sequence

\[
\left( \| (z - \mu)^r \varphi_r(z), \varphi_r(z) \| \right)_{r=0}^{N-1-r}.
\]

If all the elements are “sufficiently small,” then we conclude that indeed \( n = r \) and we stop.

The form \( \langle \cdot, \cdot \rangle \) is evaluated via numerical integration along \( \gamma \). In other words, it is approximated by a quadrature sum. We assume that this sum is calculated in the standard way, by adding the terms one by one, in other words, by forming a sequence of partial sums. We ask the quadrature algorithm not only for an approximation of the integral, say result, but also for the modulus of the partial sum that has the largest modulus, say maxsum. Then

\[
\log_{10} \frac{\text{maxsum}}{\text{result}}
\]

is an estimate for the loss of precision. This information will turn out to be extremely useful, for example in the stopping criterion.

These considerations lead to the following algorithm.

**Algorithm**

**Input** \( \langle \cdot, \cdot \rangle, \epsilon_{\text{stop}} \)

**Output** \( n, \text{zeros} \)

**Comment** \( \text{zeros} = \{ z_1, \ldots, z_n \} \). We assume that \( \epsilon_{\text{stop}} > 0 \).

\( N \leftarrow \langle 1, 1 \rangle \)

**if** \( N == 0 \) **then**

\( n \leftarrow 0; \text{zeros} \leftarrow \emptyset; \text{stop} \)

**else**

\( \varphi_0(z) \leftarrow 1 \)

\( \mu \leftarrow \langle z, 1 \rangle / N; \varphi_1(z) \leftarrow z - \mu \)

\( r \leftarrow 1; t \leftarrow 0 \)

**while** \( r + t < N \) **do**

**regular** **←** it is numerically feasible to generate \( \varphi_{r+t+1}(z) \) as a regular FOP \( \ldots [1] \)

**if** regular **then**

**generate** \( \varphi_{r+t+1}(z) \) as a regular FOP \( \ldots [2] \)

\( r \leftarrow r + t + 1; t \leftarrow 0 \)

\( \text{allsmall} \leftarrow \text{true}; r \leftarrow 0 \)
While allsmall and \((r + \tau < N)\) do

\[ [\text{i}p, \text{maxpsum}] \leftarrow \langle (z - \mu) \varphi_r(z), \varphi_r(z) \rangle \]  
\[ \text{i}p \leftarrow \text{i}p - 1 \]  
\[ \text{allsmall} \leftarrow (\text{i}p/\text{maxpsum} < \epsilon_{\text{stop}}) \]  
\[ \tau \leftarrow \tau + 1 \]
end while

If allsmall then

\[ n \leftarrow \tau; \text{zeros} \leftarrow \text{roots(}\varphi_r\text{)}; \text{stop} \]
end if

Else

\[ \text{generate } \varphi_{r+1}(z) \text{ as an inner polynomial } \]
\[ t \leftarrow t + 1 \]
end if

end while

\[ n \leftarrow N; \text{zeros} \leftarrow \text{roots(}\varphi_N\text{)}; \text{stop} \]
end if

Comments:

1. Statement [1] is crucial. But how does one decide that it is numerically feasible to generate the next polynomial in the sequence \(\varphi_0(z), \varphi_1(z), \ldots, \varphi_n(z)\) as a regular FOP? Suppose that the algorithm has just generated a regular FOP \(\varphi_r(z)\). Then (26) implies that if \(r\) is a regular index, then \(r + 1\) is a regular index if and only if \(\langle \varphi_r(z), \varphi_r(z) \rangle \neq 0\). This suggests the following criterion: if \(\langle \varphi_r(z), \varphi_r(z) \rangle/\text{maxpsum} < \epsilon_{\text{regular}}\), where \(\epsilon_{\text{regular}}\) is some small threshold given by the user, then define \(\varphi_{r+1}(z)\) as an inner polynomial, else define it as a regular FOP. However, as we will illustrate in the next section, it is very difficult to choose an appropriate value of \(\epsilon_{\text{regular}}\).

We prefer to use the following criterion: act as if the next polynomial in the sequence, say \(\varphi_t(z)\), is defined as a regular FOP, i.e., compute its zeros by computing the eigenvalues of the pencil \(G_t^1 - \lambda G_t\) and then check if these zeros lie sufficiently close to the interior of \(\gamma\). If so, then define \(\varphi_t(z)\) as a regular FOP, else define it as an inner polynomial. The idea behind this strategy is the following. If the matrix \(G_t\) is singular, in which case also the matrix \(H_t\) is singular of course, then the pencil \(G_t^1 - \lambda G_t\) has either a number of eigenvalues at infinity or a number of eigenvalues that may assume arbitrary values. Indeed, by using the structure of the matrices \(G_t^1\) and \(G_t\), one can easily prove the following result, which complements Theorem 7.

**Theorem 10.** Let \(t \geq 1\) be an integer, let \(r\) be the largest regular index less than or equal to \(t\), and let \(r'\) be the smallest regular index greater than \(t\). (Define \(r' := +\infty\) if \(t \geq n\).) Then the eigenvalues of the pencil \(G_t^1 - \lambda G_t\) are given by the eigenvalues of the pencil \(G_r^1 - \lambda G_r\) and \(t-r\) eigenvalues that may assume arbitrary values if \(t < r' - 1\) or \(t-r\) eigenvalues \(\lambda = \infty\) if \(t = r' - 1\).

Each of these indeterminate eigenvalues corresponds to two corresponding zeros on the diagonals of the generalized Schur decomposition of \(G_t^1\) and
When actually calculated, these diagonal entries are different from zero because of roundoff errors. The quotient of two such corresponding diagonal entries is a spurious eigenvalue. Our strategy is based on the assumption that, if the matrix $H_t$, and thus also the matrix $G_t$, is nearly singular, then the computed eigenvalues of the pencil $G_t^{(1)} - \lambda G_t$ that correspond to the eigenvalues that lie at infinity or that may assume arbitrary values, lie far away from the interior of $\gamma$.

The reader may object that our criterion is too strict. Indeed, the zeros of the regular FOPs of degree $n$ need not lie close to the interior of $\gamma$, except if the form $(\cdot, \cdot)$ is a true (positive definite) inner product, in which case the zeros of the regular FOPs lie in the convex hull of $\{z_1, \ldots, z_n\}$. (This follows from a general result on orthogonal polynomials. See, e.g., Van Assche [273].) Thus it may very well be the case that some of the computed zeros of a well-conditioned regular FOP lie far away from $\gamma$, in which case our algorithm decides to define this polynomial as an inner polynomial. In other words, our algorithm may define more inner polynomials than strictly necessary. However, we have done a lot of numerical tests and have found that our strategy leads to very accurate results. Also, compared to the criterion based on inner products of the type $\langle \varphi_r(z), \varphi_r(z) \rangle$, another advantage is that the user doesn’t have to supply a threshold such as $\varepsilon_{\text{regular}}$.

2. Statement [2] means: define $\varphi_{r+1}(z)$ as $\varphi_{r+1}(z) \leftarrow \prod_{j=1}^{r+1} (z - \alpha_j)$. The zeros $\alpha_j$ are computed as $\alpha_j = \mu + \lambda_j$, $j = 1, \ldots, r+1$, where $\lambda_1, \ldots, \lambda_{r+1}$ are the eigenvalues of the pencil $G_t^{(1)} - \lambda G_{r+1}$, cf. Theorem 7.

3. In statement [3] we use the inner product $\langle (z - \mu)^T \varphi_r(z), \varphi_r(z) \rangle$ and not $\langle z^T \varphi_r(z), \varphi_r(z) \rangle$ as it is likely that the former leads to more accurate results than the latter. If $r \leq \tau$, then one may also use $\langle \varphi_r(z) \varphi_r(z), \varphi_r(z) \rangle$. In general, if $\tau = \alpha r + \beta$, where $\alpha, \beta \in \mathbb{N}$ with $\beta < r$, then one may use $\langle (\varphi_r(z))^\beta, \varphi_r(z) \rangle$.

4. Observe that in statement [4] we do not compare $\mu$ with $\varepsilon_{\text{stop}}$ but take into account the loss of precision as estimated by the quadrature algorithm. We have found this heuristic to be very reliable.

5. In statement [5] one may define $\varphi_{r+1}(z)$ as $\varphi_{r+1}(z) \leftarrow (z - \mu) \varphi_{r+1}(z)$ or $\varphi_{r+1}(z) \leftarrow \varphi_{r+1}(z) \varphi_r(z)$. In our opinion, both versions are to be preferred to the “classical” $\varphi_{r+1}(z) \leftarrow z^{r+1} \varphi_r(z)$.

6. Instead of computing $\mu$, the arithmetic mean of the zeros, as $\mu \leftarrow (z, 1)/N$, one can also use the following formula, which may give a more accurate result: $\mu \leftarrow w + (z - w, 1)/N$, where $w$ is a point inside $\gamma$, preferably near the centre of the interior of $\gamma$.

7. As we represent our FOPs by using the product representation,

$$\varphi(z) = \prod_{\alpha \in \varphi^{-1}(0)} (z - \alpha),$$

the function roots($\cdot$) is obviously not a function that calculates the zeros of a polynomial from its coefficients in the standard monomial basis.
We have implemented our algorithm in Matlab (for disks) and also in Fortran 90 (for rectangular regions). The latter implementation will be presented in more detail in Section 5. Numerical examples will be given in Section 4 and also in Section 5. They will illustrate the effectiveness of our approach.

4. Numerical examples

In the following examples we have considered the case that $\gamma$ is a circle. The computations have been done via Matlab 5 (with floating point relative accuracy $\approx 2.2204 \times 10^{-16}$).

The following integration algorithm is used to approximate the form $\langle \gamma, \cdot \rangle$. Let $\gamma$ be the circle with centre $c$ and radius $\rho$. Then

$$ (\phi, \psi) = \rho \int_0^1 \phi(c + \rho e^{2\pi i \theta}) \psi(c + \rho e^{2\pi i \theta}) \frac{f(c + \rho e^{2\pi i \theta})}{f(c + \rho e^{2\pi i \theta})} e^{2\pi i \theta} d\theta. $$

Since this is the integral of a periodic function over a complete period, the trapezoidal rule is an appropriate quadrature rule. If $F : [0, 1] \to \mathbb{C}$ is the integrand in the right-hand side of (35), then the $q$-point trapezoidal rule approximation to $\langle \phi, \psi \rangle$ is given by

$$ \langle \phi, \psi \rangle = \int_0^1 F(\theta) d\theta \approx \frac{1}{q} \sum_{k=0}^{q-1} F(k/q) =: T_q. $$

The double prime indicates that the first and the last term of the sum are to be multiplied by $1/2$. As $F$ is periodic with period one, we may rewrite $T_q$ as

$$ T_q = \frac{1}{q} \sum_{k=0}^{q-1} F(k/q). $$

This shows that $T_q$ indeed depends on $q$ (and not $q + 1$) points. As

$$ T_{2q} = \frac{1}{2} T_q + T_{q\to2q} $$

where

$$ T_{q\to2q} := \frac{1}{2q} \sum_{k=0}^{q-1} F\left(\frac{2k + 1}{2q}\right), $$

successive doubling of $q$ enables us in each step to reuse the integrand values needed in the previous step. In the following examples we started with $q = 16$ and continued doubling $q$ until $|T_{2q} - T_q|$ was sufficiently small. More precisely, if $S_q$ and $S_{q\to2q}$ denote the modulus of the partial sum of $qT_q$ respectively $2qT_{q\to2q}$ that has the largest modulus, then our stopping criterion is given by $|T_{2q} - T_q| \leq S_{2q} \times 10^{-14}$, where $S_{2q} := \max\{S_q, S_{q\to2q}\}/2q$.

Lyness and Delves [203] studied the asymptotic behaviour of the quadrature error. They showed that the modulus of the error made by the $q$-point trapezoidal rule is asymptotically $O(A^q)$ where $0 \leq A < 1$. More precisely,

$$ A := \max\left\{ \frac{1}{\rho}, \frac{\rho}{\| \frac{\rho}{\rho} \|} \right\} $$
where \( z_1 \) is the zero of \( f \) that lies closest to \( \gamma \) and in the interior of \( \gamma \), \( z_E \) is the zero of \( f \) that lies closest to \( \gamma \) and in the exterior of \( \gamma \), and \( \rho_s \) is the distance between \( c \) and the nearest singularity of \( f \).

**Example 2.** Our first example illustrates the importance of shifting the origin in the complex plane to the arithmetic mean of the zeros. It also compares the two strategies that we have proposed to decide whether it is numerically feasible to generate the next polynomial in the sequence \( \varphi_0(z), \varphi_1(z), \ldots, \varphi_n(z) \) as a regular FOP. We will see that it is indeed better to act as if the polynomial is a regular FOP, i.e., to compute its zeros by solving the generalized eigenvalue problem of Theorem 7, and then to check if these zeros lie sufficiently close to the interior of \( \gamma \). If so, the polynomial is indeed defined as a regular FOP, else it is defined as an inner polynomial.

Suppose that \( n = 3, z_1 = \epsilon, z_2 = \sqrt{3} + i, z_3 = \sqrt{3} - i \), and \( \nu_1 = \nu_2 = \nu_3 = 1 \). That is, suppose that \( f(z) = (z - \epsilon)(z - \sqrt{3})^2 + 1 \). If \( \epsilon = 0 \), then the Hankel matrix \( H_2 \) is exactly singular, i.e., \( \varphi_2(z) \) has to be defined as an inner polynomial. We set \( \epsilon = 10^{-2} \). Suppose that \( \gamma = \{ z \in \mathbb{C} : |z| = 3 \} \). In the quadrature algorithm, we have evaluated the logarithmic derivative \( f'(z)/f(z) \) of \( f(z) \) via the formula

\[
\frac{f'(z)}{f(z)} = \sum_{k=1}^{n} \frac{\nu_k}{z - z_k}
\]

We have taken \( \epsilon_{stop} = 10^{-18} \). Our algorithm proceeds as follows. The total number of zeros \( N \) is equal to 3. The polynomial \( \varphi_0(z) \) is of course defined as a regular FOP, \( \varphi_0(z) \leftarrow 1 \). The arithmetic mean \( \mu \) is approximated by

1. 1.58033871706381e+00 + i 1.526556658859590e-16.

The polynomial \( \varphi_1(z) \) is also defined as a regular FOP, \( \varphi_1(z) \leftarrow z - \mu \). The inner product \( \langle \varphi_1(z), \varphi_1(z) \rangle \) is equal to 0.02303. To take into account the loss of precision, we divide by \( S_2 \) to obtain its scaled counterpart, which is equal to 0.01231. Is this that small that we should define \( \varphi_2(z) \) as an inner polynomial? It seems not, and we decide to define \( \varphi_2(z) \) as a regular FOP. Its zeros are approximated by

1. 1.58072473165547e+00 + i 1.51325865430580e-16
2. 2.000049565733722e+02 + i 3.487413819470906e-12

Note how large the second zero is! The inner product \( \langle \varphi_2(z), \varphi_2(z) \rangle \) is equal to 910.504. Its scaled counterpart is 0.01241. We decide to define \( \varphi_3(z) \) as a regular FOP. Its zeros are approximated by

1. 1.732050807571817e+00 - i 1.000000000004209e+00
2. 1.000000006616760e+02 + i 3.913802997325630e-12
3. 1.732050807566777e+00 + i 1.000000000002807e+00

As \( N = 3 \), we may stop. The relative errors of the approximations for the zeros of \( f \) are \( O(10^{-12}) \), except for the zero that approximates \( z_1 = \epsilon \), which has a relative error of \( O(10^{-10}) \). The absolute errors are \( O(10^{-12}) \). The relative errors of the approximations for the multiplicities are \( O(10^{-11}) \).

As one of the zeros of \( \varphi_2(z) \) lies far away from the interior of \( \gamma \), we should decide to define \( \varphi_2(z) \) as an inner polynomial. Surprisingly, this does not improve the accuracy of the results. However, let us see what happens if we first shift the
Example 3. Let \( f(z) = e^z + 2z \cos z - 1 \) and \( \gamma = \{z \in \mathbb{C} : |z| = 2\} \). We set \( \epsilon_{\text{stop}} = 10^{-18} \). Our algorithm finds that \( \gamma = 4 \). It defines \( \varphi_0(z) \) and \( \varphi_1(z) \) as regular FOPs. From the eigenvalues of the pencil \( G_2^{(1)} - \lambda G_2 \) it concludes that \( \varphi_2(z) \) would have a zero of modulus \( \approx 43 \) in case \( \varphi_2(z) \) is defined as a regular FOP. Thus the algorithm decides to define \( \varphi_2(z) \) as an inner polynomial. The polynomials \( \varphi_0(z) \) and \( \varphi_2(z) \) are defined as regular FOPs. Our algorithm concludes that \( n = 4 \). The computed approximations for the zeros of \( f \) are given by

\[
\begin{align*}
-1.44233953262213e+00 & - 1.106288924192872e-16 \\
5.3089430292997e-01 & + 1.331791876751121e+00 \\
5.3089430292993e-01 & - 1.331791876751121e+00 \\
-5.42337245047638e-15 & + 3.76263028319976e-16
\end{align*}
\]

The corresponding approximations for the multiplicities are

\[
\begin{align*}
1.00000000000000 \times 10^{1} & + 9.27942312879846e-17 \\
1.0000000000001 \times 10^{0} & - 2.415808667423342e-15 \\
1.0000000000001 \times 10^{0} & + 1.187431378902999e-15 \\
9.99999999999997 \times 10^{-1} & + 1.142195850770565e-15
\end{align*}
\]

By refining the approximations for the zeros of \( f \) via Newton’s method, we find that they have a relative error of \( O(10^{-16}) \), except for the approximation of \( z_4 = 0 \), which has an absolute error of \( O(10^{-15}) \). If \( \varphi_2(z) \) is defined as a regular FOP, the errors are \( O(10^{-13}) \).}

Example 4. Suppose that \( f(z) = z^2(z - 1)(z - 2)(z - 3)(z - 4) + z \sin z \) and \( \gamma = \{z \in \mathbb{C} : |z| = 5\} \). Note that \( f \) has a double zero at the origin. Our algorithm finds that \( \gamma = 6 \). It defines \( \varphi_0(z) \), \( \varphi_1(z) \), \( \varphi_2(z) \), \( \varphi_3(z) \), \( \varphi_4(z) \) and \( \varphi_5(z) \) as regular FOPs. For \( k = 2, 3, 4 \) and 5, the scaled counterparts of \( \{\varphi_k(z), \varphi_k(z)\} \) are given by

\[
\begin{align*}
7.040331724952680e-02 \\
1.910625118197985e-03 \\
1.23657574513765e-05 \\
2.42651768477941e-15
\end{align*}
\]

If we set \( \epsilon_{\text{stop}} \) to a value that is larger than \( 2.510^{-15} \), then the algorithm stops. It decides (correctly) that \( n = 5 \). The computed approximations for the zeros of \( f \) are given by

\[
\begin{align*}
2.853939307104127e-12 & + 1.218636779565894e-12 \\
1.18906588993786e+00 & + 1.174992347177040e-10 \\
1.728434986506658e+00 & + 1.58763697134256e-10 \\
3.01990732813121e+00 & + 1.7575497887398162e-11 \\
4.030381916062330e+00 & + 7.769722658005202e-13
\end{align*}
\]
The corresponding approximations for the multiplicities are:

\[
\begin{align*}
2.00000000021947e+00 + i & 9.312570667094433e-12 \\
1.000000000957614e+00 + i & 4.35256036662006e-10 \\
9.99999999867526e-01 - i & 4.117654248790392e-10 \\
9.99999999418555e-01 - i & 2.906361792434579e-11 \\
9.9999999918315e-01 - i & 3.739564528915041e-12 
\end{align*}
\]

By refining the approximations for the zeros iteratively via Newton’s method, we find that the absolute errors are \(O(10^{-12}), O(10^{-12}), O(10^{-12}), O(10^{-14})\) and \(O(10^{-16})\), respectively. The refined approximations for the zeros are given by:

\[
\begin{align*}
2.853939307034829e-12 + i & 1.218663779496334e-12 \\
1.189065890003747e+00 + i & 1.218663825423787e-10 \\
1.728434986305392e+00 + i & 1.57338495073740e-10 \\
3.019907328131241e+00 + i & 1.759027974661844e-11 \\
4.030381916062330e+00 + i & 7.768170803990681e-13
\end{align*}
\]

If we set \(\epsilon_{\text{stop}}\) to a value that is smaller than \(2.4 \times 10^{-15}\), then the algorithm continues. It defines \(\varphi_0(z)\) as a regular F OP and stops. The computed approximations for the zeros of \(f\) are now given by:

\[
\begin{align*}
-3.412647942013791e-11 - i & 3.671305864695379e-12 \\
1.189065887181205e+00 - i & 7.212695548387926e-11 \\
1.7284349863085149e+00 - i & 2.16255731479441e-11 \\
3.019907327801388e+00 + i & 1.210604626933557e-11 \\
4.030381916045125e+00 + i & 1.485878696458235e-12 \\
-2.853807137641358e+00 + i & 1.433406287105723e+00
\end{align*}
\]

whereas the corresponding approximations for the multiplicities are:

\[
\begin{align*}
1.99999999750989e-11 - i & 1.9828896140226239e-11 \\
1.999999811973171e-01 - i & 1.399505295104063e-10 \\
1.0000000008572735e+00 + i & 1.723707922538161e-10 \\
1.000000000489271e+00 - i & 7.095346599149408e-12 \\
1.00000000067249e+00 - i & 5.486513691338396e-12 \\
2.023577344691327e-14 - i & 9.441050659615426e-15
\end{align*}
\]

Observe that we find more or less the same approximations for the zeros as before and also a spurious “zero,” cf. Theorem 10. Fortunately, the presence of spurious zeros can be easily detected, as their corresponding “multiplicities” are equal to zero. This can be explained as follows. The Vandermonde matrix that corresponds to the calculated approximations for the zeros will almost surely (i.e., with probability one) be nonsingular and therefore the system for the multiplicities will have only one solution, which gives the true zeros of \(f\) their correct corresponding multiplicity and the spurious ones multiplicity zero. \(\Diamond\)

**Example 5.** Let \(f(z) = z^2(z - 2)^2 [e^{2z} \cos z + z^3 - 1 - \sin z]\) and \(\gamma = \{z \in \mathbb{C} : |z| = 3\}\). Note that \(f\) has a triple zero at the origin and a double zero at \(z = 2\). Our algorithm finds that \(N = 8\). It defines \(\varphi_0(z), \varphi_1(z), \varphi_2(z), \varphi_3(z), \varphi_4(z)\) and \(\varphi_5(z)\) as regular F OPs. For \(k = 2, 3, 4\) and 5, the scaled counterparts of \(|\varphi_k(z), \varphi_k(z)|\) are given by...
If we set \( \varepsilon_{\text{stop}} \) to a value that is larger than \( 5 \times 10^{-15} \), then the algorithm stops. It decides (correctly) that \( n = 5 \). The computed approximations for the zeros of \( f \) are given by

\[
\begin{align*}
-4.607141197276816 \times 10^{-01} &+ i 6.254277693477422 \times 10^{-01} \\
-4.607141197280875 \times 10^{-01} &- i 6.254277693470294 \times 10^{-01} \\
2.65332006551662 \times 10^{-12} &+ i 8.232702952841388 \times 10^{-13} \\
2.00000000000728 \times 10^+00 &+ i 2.13363838712091 \times 10^{-13} \\
1.664682869749229 \times 10^+00 &+ i 9.605975127915706 \times 10^{-13}
\end{align*}
\]

The corresponding approximations for the multiplicities are

\[
\begin{align*}
1.000000000003826 \times 10^+00 &+ i 5.208513390226470 \times 10^{-12} \\
1.000000000005729 \times 10^+00 &- i 1.653703539980123 \times 10^{-12} \\
2.9999999991895 \times 10^+00 &- i 3.202939915103751 \times 10^{-12} \\
1.9999999997495 \times 10^+00 &- i 3.449339571824102 \times 10^{-12} \\
1.00000000011054 \times 10^+00 &+ i 3.097374215634804 \times 10^{-12}
\end{align*}
\]

By refining the approximations for the zeros iteratively via Newton's method, we find that the absolute errors are \( O(10^{-14}) \), \( O(10^{-12}) \), \( O(10^{-12}) \), \( O(10^{-13}) \) and \( O(10^{-12}) \), respectively.

If we set \( \varepsilon_{\text{stop}} \) to a value that is smaller than \( 10^{-15} \), then the algorithm continues until the very end. For \( k = 6 \) and \( 7 \), the scaled counterparts of \( |(\varphi_k(z), \varphi_k(z))| \) are given by

\[
\begin{align*}
3.073186423911271 \times 10^{-15} \\
1.658776045256565 \times 10^{-15}
\end{align*}
\]

The algorithm defines \( \varphi_6(z) \) and \( \varphi_7(z) \) as regular FOPs. Then it defines \( \varphi_8(z) \) as an inner polynomial and stops. The computed approximations for the zeros of \( f \) are now given by

\[
\begin{align*}
-4.607141197287676 \times 10^{-01} &+ i 6.254277693479825 \times 10^{-01} \\
-4.607141197287121 \times 10^{-01} &- i 6.254277693478455 \times 10^{-01} \\
6.096234628216735 \times 10^{-13} &+ i 4.562446978887860 \times 10^{-14} \\
2.00000000000117 \times 10^+00 &- i 2.873008259688547 \times 10^{-13} \\
1.664682869746416 \times 10^+00 &- i 1.002467422184267 \times 10^{-12} \\
-2.767401251474829 \times 10^+00 &+ i 1.339901958409356 \times 10^+00 \\
1.042158257408570 \times 10^+00 &- i 3.11040305104297 \times 10^+00 \\
5.929068287859467 \times 10^{-01} &- i 9.822871682718670 \times 10^{-17}
\end{align*}
\]

whereas the corresponding approximations for the multiplicities are

\[
\begin{align*}
1.00000000000205 \times 10^+00 &+ i 1.226534018772486 \times 10^{-12} \\
1.00000000000592 \times 10^+00 &- i 1.131203882923728 \times 10^{-12} \\
2.9999999999770 \times 10^+00 &- i 2.952132467539878 \times 10^{-13} \\
1.9999999999756 \times 10^+00 &+ i 4.32589706507562 \times 10^{-12} \\
1.00000000001872 \times 10^+00 &- i 4.180646412170987 \times 10^{-12}
\end{align*}
\]
As discussed in Example 4, the approximations for the multiplicities enable us to locate spurious zeros.

\textbf{Example 6.} Let us consider the Wilkinson polynomial of degree 10, \( f(z) = \prod_{k=1}^{10} (z - k) \). Suppose that \( \gamma = \{ z \in \mathbb{C} : |z| = 11 \} \). We have evaluated the logarithmic derivative of \( f \) via the formula (36). By using Theorem 4 we obtain the following approximations for the zeros of \( f \):

\[
\begin{align*}
1.000069039770840e+00 & \quad -i \quad 6.902850482141518e-05 \\
2.004976266975782e+00 & \quad -i \quad 3.428732045109014e-03 \\
3.069632060117111e+00 & \quad -i \quad 2.595941486831997e-02 \\
4.251792296513047e+00 & \quad -i \quad 6.062429488533006e-02 \\
5.546923068519217e+00 & \quad -i \quad 6.702269982182261e-02 \\
6.815527607409872e+00 & \quad -i \quad 4.182297169281859e-02 \\
7.96066732397854e+00 & \quad -i \quad 1.296711240552063e-02 \\
8.997081596754466e+00 & \quad -i \quad 1.314415641505938e-03 \\
9.99960580756190e+00 & \quad -i \quad 2.20763616991653e-05 \\
3.755000277916281e+00 & \quad -i \quad 4.101263665758313e+00
\end{align*}
\]

The absolute errors of the approximations for the multiplicities are \( O(10^{-1}) \).

Now let us see how our algorithm performs. We set \( \epsilon_{\text{stop}} = 10^{-18} \). It finds that \( N = 10 \). It defines \( \varphi_0(z), \varphi_1(z), \ldots, \varphi_9(z) \) as regular FOPs. For \( k = 2, 3, \ldots, 9 \), the scaled counterparts of \(|\langle \varphi_k(z), \varphi_k(z) \rangle|\) are given by

\[
\begin{align*}
3.816240840818399e-02 \\
1.299886508906636e-03 \\
3.553824638723713e-05 \\
8.03969836655619e-07 \\
1.469066868160238e-08 \\
2.01006390239241e-10 \\
1.220433804961808e-12 \\
9.741759723349369e-16
\end{align*}
\]

This clearly indicates that the problem is very ill-conditioned. The algorithm defines \( \varphi_{10}(z) \) as an inner polynomial and stops. The relative errors of the approximations for the zeros are given by

\[
\begin{align*}
1.005577196493408e-04 \\
2.863610448143865e-03 \\
2.00971359668208e-02 \\
5.876683150395577e-02 \\
9.9999999999999e-02 \\
8.14185770057402e-02 \\
3.100882100301538e-02 \\
6.60936435112427e-03 \\
5.20783593342222e-04 \\
7.598125455914798e-06
\end{align*}
\]
respectively. However, if we consider the circle $\gamma = \{ z \in \mathbb{C} : |z - 5.5| = 5.5 \}$, then the following happens. The algorithm defines only regular FOPs. For $k = 2, 3, \ldots, 9$, the scaled counterparts of $|\varphi_k(z), \varphi_k(z)|$ are now given by

1. $1.120889480330385 \times 10^0$
2. $3.130941249560411 \times 10^0$
3. $6.391288244829463 \times 10^0$
4. $9.005452570225671 \times 10^0$
5. $1.142273634328999 \times 10^0$
6. $1.221288814387772 \times 10^0$
7. $2.425547309622016 \times 10^0$
8. $2.149861073624070 \times 10^0$

The relative errors of the approximations for the zeros are given by

1. $8.362431354624777 \times 10^{-12}$
2. $4.684075580626331 \times 10^{-10}$
3. $6.568404577352624 \times 10^{-9}$
4. $3.252872101214176 \times 10^{-8}$
5. $6.598954186243715 \times 10^{-8}$
6. $5.80079386415380 \times 10^{-8}$
7. $2.19653648456049 \times 10^{-8}$
8. $3.331673425077182 \times 10^{-8}$
9. $1.672314153254196 \times 10^{-10}$
10. $1.65104535618564 \times 10^{-12}$

whereas the absolute errors of the approximations for the multiplicities are

1. $4.518653295430768 \times 10^{-11}$
2. $3.537203913804699 \times 10^{-9}$
3. $4.76370740887607 \times 10^{-8}$
4. $1.8088181111186228 \times 10^{-7}$
5. $1.605226012463829 \times 10^{-7}$
6. $1.31687759500569 \times 10^{-7}$
7. $1.948007501440231 \times 10^{-7}$
8. $6.042232266569582 \times 10^{-8}$
9. $5.351870471708607 \times 10^{-9}$
10. $9.475094893959826 \times 10^{-11}$

**Example 7.** Recently, Engelsbohrs, Luzyanina and Roose [75] have used our algorithm to compute all the zeros of

$$f(z) = a + bz + z^2 - h z^2 e^{-\tau z},$$

where $a = 1, b = 0.5, h = -0.82465048736655$ and $\tau = 6.74469732735569$, that lie in the rectangular region

$$\{ z \in \mathbb{C} : 0.3 \leq \text{Re} z \leq 0.1, \quad -24.7 \leq \text{Im} z \leq 24.7 \}.$$ 

The results are shown in Figure 3.

These zeros determine the stability of a steady state solution of a neutral functional differential equation. For the given values of the parameters $a$, $b$, $h$ and $\tau$, this steady
Figure 3. Zeros of $a + bz + z^2 + hz^2e^{-rz}$ where $a = 1$, $b = 0.5$, $h = -0.82465048736655$ and $r = 6.744969732735569$.

state solution is also a Hopf bifurcation point. The authors have transformed the computed zeros into the Floquet multipliers of the corresponding emanating periodic solution. This has enabled them to compare these “exact” Floquet multipliers with the approximations obtained via their approach.

Note. As our algorithm not only obtains approximations for the zeros but also the corresponding multiplicities, we can use the modified Newton’s method

$$z_k^{(a+1)} = z_k^{(a)} - \nu_k \frac{f(z_k^{(a)})}{f'(z_k^{(a)})}, \quad \alpha = 0, 1, 2, \ldots,$$

to refine the approximations for the zeros. The modified Newton’s method, Van de Vel’s related iteration and our multidimensional generalizations of these methods will be discussed in detail in Chapter 7.

More numerical examples will be given in Section 5 and also in Chapter 2.

5. The software package ZEAL

We have implemented the algorithm that we have presented in Section 3 for the case that the curve $\gamma$ is a rectangle whose edges are parallel to the coordinate axes. Our package is called ZEAL (“ZEros of Analytic functions”) and is written in Fortran 90. Numerical approximations for the integrals along $\gamma$ are computed via the quadrature package QUADPACK [241].

ZEAL’s user interface is inspired by that of our package ZEBEC [185], which we will present in Chapter 5. Once the user has specified the rectangle $\gamma$, the analytic function $f$, its first derivative $f'$ and the value of the parameter $M$ (i.e., the number of zeros that are to be calculated simultaneously, cf. our discussion of the method of Delvès and Lyness in Section 1), he/she can ask ZEAL to compute only the total number of zeros of $f$ that lie inside $\gamma$, to isolate subregions of the interior of $\gamma$ that contain at most $M$ zeros, to compute all the zeros of $f$ that lie inside $\gamma$ or to compute
only a specified number of zeros (together with their respective multiplicities). The results can be written to separate files, etc. All these options will be discussed in detail below.

5.1. The approach taken by ZEAL. Given a rectangle whose edges are parallel to the coordinate axes and a positive integer $M$, we take the following approach:

- We calculate the total number of zeros that lie inside this rectangle.
- Via consecutive subdivisions we obtain a set of subrectangles, each of which contains at most $M$ zeros (counting multiplicities).
- For each of these subrectangles, we calculate approximations for the zeros that lie inside it, together with their respective multiplicities.
- The approximations for the zeros are refined iteratively via the modified Newton's method.

As the function $f$ may have zeros on the boundary of the rectangular box specified by the user, ZEAL starts by perturbing this box. For this purpose a tolerance is used that is taken to be proportional to a power of the machine precision, for example 10 times the square root of the machine precision. The box is then slightly enlarged in an asymmetrical way. The reason for this asymmetric perturbation is to lower the possibility of having a zero close to or on any boundary of the consecutive subdivisions (see also the footnote below). For example, if the starting box is symmetric with respect to the imaginary axis, then the inner boundary at the first subdivision will pass through any imaginary zeros of $f$.

The total number of zeros of $f$ that lie inside the perturbed box is obtained in the same way as in our package ZEBEC. The real part of the integral in (12) is written as a sum of four integrals, where each integral corresponds to one of the edges of the rectangular region. Details will be given in Chapter 5. Approximations for these integrals are calculated via the adaptive integrator DQAG from QUADPACK. A zero near one of the edges of the rectangle causes the integrand of the corresponding integral to have a "peak." The closer the zero lies to the edge, the sharper this peak is. If the zero lies on the edge, then the integral is divergent. DQAG uses adaptive strategies that enable it to cope with such peaks efficiently. However, if a zero lies too close to an edge (the corresponding peak is too sharp), then DQAG warns us that it has problems in calculating the integral. Our algorithm then slightly moves this edge and restarts. By enlarging the user's box, we may of course include additional zeros. We have decided not to discard any of these zeros ourselves. Rather, we provide the user with the box that eventually has been considered, all the zeros that lie inside this box, and leave it to him/her to filter out unwanted zeros.

If the starting box (as perturbed by ZEAL) contains less than $M$ zeros (counting multiplicities), then approximations for these zeros and the values of their respective multiplicities are computed via our implementation of the algorithm that we have presented in Section 3. Otherwise, the longest edges of the box are halved and the box is subdivided into two equal boxes. The number of zeros in each of these boxes is calculated via numerical integration. If DQAG detects a zero near the inner edge, then this edge is shifted, a process that results in an asymmetric subdivision of the
Then the two smaller boxes are examined. A box that does not contain any zero is abandoned. If a box contains less than $M$ zeros, then approximations for these zeros are calculated, together with their respective multiplicities. A box that contains more than $M$ zeros is subdivided again. This process is repeated until a set of boxes has been found, each of which contains at most $M$ zeros, and approximations for all these zeros as well as the values of their respective multiplicities have been computed. The approximations for the zeros are then refined via the modified Newton’s method, which takes into account the multiplicity of a zero and converges quadratically.

5.2. The structure of ZEAL. The package ZEAL contains about 6500 lines of code including comments. It is written in Fortran 90 and has been thoroughly tested on various UNIX machines (HP 9000 B160L, IBM RS6000 7012 and SUN SPARC Ultra-2 m1170).

ZEAL consists of 8 files, namely, the main program Main and the modules

- ZEAL_Module, which contains the main subroutine ZEAL and also the subroutine CHECK_INPUT,
- Zeros_Module, which contains the subroutines APPROXIMATE and INPROD,
- Refine_Module, which contains the subroutines REFINES and NEWTON,
- Split, which contains the subroutines INBOX and SPLITBOX,
- and finally the modules Input, Error and Quad.

ZEAL also requires the subroutine DQAG from the package QUADPACK [241] and a number of subroutines from the BLAS and LAPACK libraries [12].

The user can specify the values of the various input parameters by editing the module Input. This will be discussed in Subsection 5.3.

The main program Main has the following form:

```fortran
PROGRAM Main
  USE Input
  USE ZEAL_Module
  IMPLICIT NONE
  INTEGER :: TOTALNUMBER, DISTINCTNUMBER, REFINEDNUMBER
  INTEGER, DIMENSION(:,), POINTER :: MULTIPLICITIES
  LOGICAL, DIMENSION(:,), POINTER :: REFINEMENT_OK
  COMPLEX(KIND=DP), DIMENSION(:,), POINTER :: ZEROS, FZEROS
  CALL ZEAL(TOTALNUMBER, DISTINCTNUMBER, ZEROS, FZEROS, &
             MULTIPLICITIES, REFINEDNUMBER, REFINEMENT_OK)
END PROGRAM Main
```

1QUADPACK uses certain heuristic strategies. They work very well but nevertheless can fail. For example, we have observed that in case the function has several zeros very close (say, at a distance less than $10^{-7}$) to the boundary, then DQAG may compute the wrong total number of zeros without giving any warning message. Therefore, we advise the user to make sure (as much as possible) that the zeros are not symmetrical with respect to the boundary. In particular, as many functions have zeros on the coordinate axes, we suggest to consider boxes that are asymmetrical with respect to these axes.
Main uses the information specified by the user in Input and calls the main subroutine ZEAL. This subroutine returns the total number of zeros of the given function that lie inside the given rectangular region, the number of mutually distinct zeros, the refined approximations for the zeros and the values that the function takes at these points, the corresponding multiplicities, the number of approximations for the zeros (as computed by the subroutine APPROXIMATE) that ZEAL has been able to refine successfully via the modified Newton’s method, and finally for each computed zero a logical variable that indicates whether this refinement procedure has been successful or not.

In the design of ZEAL we have followed the recommendations for precision level maintenance described by Buckley [44]. The parameter DP that appears in the declaration of the variables ZEROS and FZEROS is defined in Input,

\[
\text{INTEGER, PARAMETER :: DP = SELECTED_REAL_KIND(15,70)}
\]

It determines the precision to which all the floating point calculations are to be done. On the computers that we have used, its current value corresponds to Fortran 77’s DOUBLE PRECISION.\(^2\)

Let us briefly describe the various parts of ZEAL.

The subroutine INBOX calculates the total number of zeros that lie inside the rectangular box given by the user. If some of the zeros lie too close to the boundary of this box and the quadrature routine DQAG fails, then INBOX perturbs the box slightly and enlarges it.

The subroutine SPLITBOX takes a box and splits it into two boxes. A symmetric splitting, which proceeds by halving the longest edges, is tried first. If the calculation of the integral along the inner edge fails, then it is assumed that some of the zeros lie too close to this edge and the inner edge is shifted.

The subroutine APPROXIMATE contains our implementation of the algorithm that we have presented in Section 3. The symmetric bilinear form (23) is evaluated via the subroutine INPROD.

The subroutine NEWTON contains our implementation of the modified Newton’s method. The subroutine REFINE calls NEWTON to refine the approximations for the zeros that APPROXIMATE has computed. If NEWTON fails, then REFINE tries again from a nearby point. If after eight attempts NEWTON still fails, then REFINE indicates that it has been unable to refine the given approximation successfully.

The subroutine ZEAL forms the main part of the package. ZEAL starts by calling CHECK_INPUT to check if the input parameters specified by the user are proper. Next it calls INBOX. If there are no zeros inside the user’s box, then the program stops. If there are less than \(M\) zeros inside the box (where the value of \(M\) can be specified in Input), then APPROXIMATE and REFINE are called. Else, the box is given

---

\(^2\)This observation is important for the following reason. As documented in its makefile, ZEAL uses certain Fortran 77 routines from QUADPACK, BLAS and LAPACK. To enable the user to compile the necessary routines from BLAS and LAPACK in case these libraries are not available on his/her computer system, we have included them with our distribution of ZEAL. However, we have included only the DOUBLE PRECISION version of these routines and hence they should be replaced by the corresponding SINGLE PRECISION routines in case a change to DP requires this. The same holds for the subroutine DQAG from QUADPACK.
to SPLITBOX. The two boxes returned by SPLITBOX are examined. A box that does not contain any zero is abandoned. A box that contains less than $M$ zeros is given to APPROXIMATE and REFINE. A box that contains more than $M$ zeros is put in a list. Then ZEAL takes the next box from this list and calls SPLITBOX. This procedure is repeated until all the zeros have been computed, together with their respective multiplicities.

The program execution terminates normally after the completion of its task. This type of termination is indicated by the value 1 of the variable INFO, which is a global variable declared in the module Error. If the value of this parameter is different from 1, then the termination of the program is abnormal. The cases of abnormal termination are the following:

- INFO=0: Improper input parameters.
- INFO=2: The procedure for the calculation of the total number of zeros has failed.
- INFO=3: The procedure for the isolation of the zeros has failed.
- INFO=4: The procedure for the computation of the zeros has failed.

This concludes our discussion of the structure of ZEAL.

5.3. ZEAL’s user interface. In the module Input the following parameters have to be set:

- LV: a real array of length 2 that contains the $x$- and $y$-coordinates of the left lower vertex of the rectangle that is to be examined.
- H: a real array of length 2 that specifies the size of this rectangle along the $x$- and $y$-direction.
- M: an integer that determines the maximum number of zeros (counting multiplicities) that are considered within a subrectangle. $M$ has to be larger than the maximum of the multiplicities of the zeros.
- ICON: an integer in $\{1, \ldots, 4\}$ that specifies which calculations are to be done:
  1. calculation of the total number of zeros, only,
  2. calculation of the total number of zeros and isolation of a set of sub-rectangles, each of which contains at most $M$ zeros,
  3. calculation of the total number of zeros and computation of all the zeros, together with their respective multiplicities,
  4. calculation of the total number of zeros and computation of $NR$ zeros, together with their respective multiplicities.

Note that if ICON=4, the user must also supply the desired number of zeros $NR$. In the other cases (ICON=1,2,3) a value of $NR$ may be supplied but it will not be used by the package.

- NUMABS: a real variable that determines the absolute accuracy to which the integrals that calculate the number of zeros are to be evaluated. In case NUMABS = 0.0_DP, only a relative criterion is used.
- NUMREL: a real variable that determines the relative accuracy to which the integrals that calculate the number of zeros are to be evaluated. In case NUMREL = 0.0_DP, only an absolute criterion is used.
If NUMABS and NUMREL are both too small, then the numerical integration may be time-consuming. If they are both too large, then the calculated number of zeros may be wrong. The default values of NUMABS and NUMREL are 0.07_DP and 0.0_DP, respectively.

INTABS: a real variable that determines the absolute accuracy to which the integrals that are used to compute approximations for the zeros are to be calculated. If INTABS = 0.0_DP, then only a relative criterion will be used.

INTREL: a real variable that determines the relative accuracy to which the integrals that are used to compute approximations for the zeros are to be calculated. If INTREL = 0.0_DP, then only an absolute criterion will be used.

If INTABS and INTREL are both too small, then the numerical integration may be time-consuming. If they are both too large, then the approximations for the zeros may be very inaccurate and Newton's method, which is used to refine these approximations (see NEWTONZ and NEWTONF), may fail. The default values of INTABS and INTREL are 0.0_DP and 1.0E-12_DP, respectively.

EPS_STOP: a real variable that is used in the stopping criterion that determines the number of mutually distinct zeros. If EPS_STOP is too large, then the computed number of mutually distinct zeros may be too small. If EPS_STOP is too small, then the computed number of mutually distinct zeros may be too large, especially in case the function has many multiple zeros. A recommended value is 1.0E-08_DP.

NEWTONZ and NEWTONF: these real variables should be specified in case ICON = 3 or 4. They are used as follows. The modified Newton's method, which takes into account the multiplicity of a zero and converges quadratically, is used to refine the calculated approximations for the zeros. The iteration stops if the relative distance between two successive approximations is at most NEWTONZ or the absolute value of the function at the last approximation is at most NEWTONF or if a maximum number of iterations (say, 20) is exceeded.

VERBOSE: a logical variable. ZEAL is allowed to print information (concerning the user's input and the computed results) if and only if VERBOSE is equal to .TRUE.

FILES: a logical variable. If FILES is set equal to .TRUE., then ZEAL generates the files zeros.dat and mult.dat. They contain the computed approximations for the zeros as well as their respective multiplicities. ZEAL also writes the file fzeros.dat, which contains the values that the function takes at the computed approximations for the zeros.

IFAIL: an integer that determines how errors are to be handled. We follow the NAG convention:
1. soft silent error—control is returned to the calling program.
-1. soft noisy error—an error message is printed and control is returned to the calling program.
0. hard noisy error—an error message is printed and the program is stopped.
In the module Input, the user also has to specify the function $f$ whose zeros ZEAL has to compute as well as its first derivative $f'$. This is done via the subroutine FDF, which takes the following form:

```fortran
SUBROUTINE FDF(Z,F,DF)

COMPLEX(KIND=DP), INTENT(IN) :: Z
COMPLEX(KIND=DP), INTENT(OUT) :: F, DF

F = ...
DF = ...

END SUBROUTINE FDF
```

The user also has to specify the logical function VALREG. Given a rectangular region specified by its left lower vertex and the sizes of its edges, VALREG decides whether the function $f$ is analytic inside this region or not. VALREG has the following form:

```fortran
FUNCTION VALREG(LV,H)

LOGICAL VALREG
REAL(KIND=DP), INTENT(IN) :: LV(2), H(2)

VALREG = ...

END FUNCTION VALREG
```

For example, if $f$ is analytic in the entire complex plane, then one may use the statement

```fortran
VALREG = .TRUE.
```

If $f$ has a branch cut along the non-positive real axis, then one may write

```fortran
VALREG = .NOT. ( LV(2)*LV(2)+H(2) <= 0.0_DP .AND. & LV(1) <= 0.0_DP )
```

This concludes our discussion of ZEAL’s user interface.

### 5.4. A few examples of how to use ZEAL

We will now discuss a few numerical examples.

**Example 8.** Suppose that $f(z) = e^z + 2z \cos z - 1$ and that

$$W = \{ z \in \mathbb{C} : -2 \leq \text{Re} z \leq 2, \quad -2 \leq \text{Im} z \leq 3 \}.$$  

In other words, $W$ is the rectangular region $[-2,2] \times [-2,3]$. Therefore, we have to define the input parameters LV and H as

```fortran
LV = (-2.0_DP,-2.0_DP/) and H = (4.0_DP, 5.0_DP/).
```

We set $M = 5$. The logical variables VERBOSE and FILES are set to .TRUE. We start by calculating only the total number of zeros, ICON = 1. ZEAL outputs the following.
This is ZEAL. Version of October 1998.

Input:

<table>
<thead>
<tr>
<th>LV</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2.000000000000000</td>
<td>-2.000000000000000</td>
</tr>
<tr>
<td>4.000000000000000</td>
<td>5.000000000000000</td>
</tr>
</tbody>
</table>

M = 5
ICON = 1
FILES = T

Results:

The following box has been considered:

<table>
<thead>
<tr>
<th>LV</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2.000000016391277</td>
<td>-2.00000019371510</td>
</tr>
<tr>
<td>4.000000035762787</td>
<td>5.000000041723251</td>
</tr>
</tbody>
</table>

Total number of zeros inside this box = 4
The function has four zeros inside the given box. We now ask ZEAL to compute approximations for all these zeros, ICON=3.

This is ZEAL. Version of October 1998.

Input:

<table>
<thead>
<tr>
<th>LV</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2.000000000000000</td>
<td>-2.000000000000000</td>
</tr>
<tr>
<td>4.000000000000000</td>
<td>5.000000000000000</td>
</tr>
</tbody>
</table>

M = 5
ICON = 3
FILES = T

Results:

The following box has been considered:

<table>
<thead>
<tr>
<th>LV</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2.000000016391277</td>
<td>-2.00000019371510</td>
</tr>
<tr>
<td>4.000000035762787</td>
<td>5.000000041723251</td>
</tr>
</tbody>
</table>

Total number of zeros inside this box = 4
Number of boxes containing at most 5 zeros = 1

These boxes are given by:
1) $LV = -2.00000016391277 \quad -2.00000019371510$

$H = 4.00000035762787 \quad 5.00000041723251$

Total number of zeros inside this box = 4

Final approximations for the zeros and verification:

1) Number of mutually distinct zeros = 4

$z = \begin{pmatrix} -1.84423395326221 \\ -0.729696337329436E-29 \end{pmatrix}$

$f(z) = \begin{pmatrix} 0.222044604925031E-15 \\ 0.297690930716218E-28 \end{pmatrix}$

multiplicity = 1

$z = \begin{pmatrix} 0.530894930292930 \\ 1.33179187675112 \end{pmatrix}$

$f(z) = \begin{pmatrix} 0.888178419700125E-15 \\ 0.222044604925031E-14 \end{pmatrix}$

multiplicity = 1

$z = \begin{pmatrix} 0.530894930292930 \\ -1.33179187675112 \end{pmatrix}$

$f(z) = \begin{pmatrix} -0.266453525910038E-14 \\ -0.44408920985063E-15 \end{pmatrix}$

multiplicity = 1

$z = \begin{pmatrix} 0.277555756299546E-16 \\ 0.732694008769276E-26 \end{pmatrix}$

$f(z) = \begin{pmatrix} 0.000000000000000 \\ 0.366347004384638E-25 \end{pmatrix}$

multiplicity = 1

If we set $M = 2$, then ZEAL outputs the following.

This is ZEAL. Version of October 1998.

Input:

$LV = -2.000000000000000 \quad -2.000000000000000$

$H = 4.000000000000000 \quad 5.000000000000000$

$M = 2$

ICON = 3

FILES = T

Results:

The following box has been considered:

$LV = -2.00000016391277 \quad -2.00000019371510$

$H = 4.00000035762787 \quad 5.00000041723251$

Total number of zeros inside this box = 4
Number of boxes containing at most 2 zeros = 3

These boxes are given by:

1) \( LV = -2.00000016391277 \) \( H = 4.00000035762787 \)  
   Total number of zeros inside this box = 1

2) \( LV = -2.00000016391277 \) \( H = 2.00000017881393 \)  
   Total number of zeros inside this box = 2

3) \( LV = 0.149011611938477E-07 \) \( H = 2.00000017881393 \)  
   Total number of zeros inside this box = 1

Final approximations for the zeros and verification:

1) Number of mutually distinct zeros = 1
   \( z = ( 0.530894930292931 , 1.33179187675112 ) \)
   \( f(z) = ( 0.888178419700125E-15,-0.17763568390025E-14 ) \)
   multiplicity = 1

2) Number of mutually distinct zeros = 2
   \( z = ( -1.84423395326221,-0.551251254781237E-21 ) \)
   \( f(z) = ( 0.2220464925031E-15, 0.224891493487409E-20 ) \)
   multiplicity = 1

   \( z = ( -0.501336236251204E-20,-0.135361644895767E-20 ) \)
   \( f(z) = ( 0.000000000000000,-0.676808224478837E-20 ) \)
   multiplicity = 1

3) Number of mutually distinct zeros = 1
   \( z = ( 0.530894930292931 , -1.33179187675112 ) \)
   \( f(z) = ( 0.888178419700125E-15,-0.444089209850063E-15 ) \)
   multiplicity = 1

Finally, suppose that we want ZEAL to compute only two zeros. We set ICON=4 and NR=2.

This is ZEAL. Version of October 1998.

Input:

\( LV = -2.00000000000000 \) \( -2.00000000000000 \)
Results:

The following box has been considered:

\[\begin{array}{cc}
LV = -2.00000016391277 & -2.00000019371510 \\
H = 4.00000035762787 & 5.00000041723251 \\
\end{array}\]

Total number of zeros inside this box = 4

Number of boxes containing at most 2 zeros = 3

These boxes are given by:

1) \[\begin{array}{cc}
LV = -2.00000016391277 & 0.500000014901161 \\
H = 4.00000035762787 & 2.50000020861626 \\
\end{array}\]

Total number of zeros inside this box = 1

2) \[\begin{array}{cc}
LV = -2.00000016391277 & -2.00000019371510 \\
H = 2.00000017881393 & 2.50000020861626 \\
\end{array}\]

Total number of zeros inside this box = 2

3) \[\begin{array}{cc}
LV = 0.149011611938477E-07 & -2.00000019371510 \\
H = 2.00000017881393 & 2.50000020861626 \\
\end{array}\]

Total number of zeros inside this box = 1

Requested number of mutually distinct zeros = 2

Final approximations for the zeros and verification:

1) Number of mutually distinct zeros = 1

\[\begin{array}{c}
z = ( 0.530894930292931, 1.33179187675112 ) \\
f(z) = ( 0.888178419700125E-15, -0.177635683940025E-14 ) \\
multiplicity = 1 \\
\end{array}\]

2) Number of mutually distinct zeros = 2

\[\begin{array}{c}
z = ( -1.84423395326221, -0.551251254781237E-21 ) \\
\end{array}\]
**Example 9.** Suppose that \( f(z) = z^2(z - 1)(z - 2)(z - 3)(z - 4) + z\sin z \) and let \( W \) be the rectangular region determined by
\[
\text{LV} = (-0.5\_DP, -0.5\_DP) \quad \text{and} \quad \text{H} = (6.0\_DP, 2.0\_DP).
\]
Note that \( f \) has a double zero at the origin. We set \( M=5 \) and \( \text{ICON}=3 \).

This is ZEAL. Version of October 1998.

Input:

\[
\begin{aligned}
\text{LV} &= \quad -0.5000000000000000 \quad -0.5000000000000000 \\
\text{H} &= \quad 6.0000000000000000 \quad 2.0000000000000000 \\
M &= \quad 5 \\
\text{ICON} &= \quad 3 \\
\text{FILES} &= \quad T
\end{aligned}
\]

Results:

The following box has been considered:

\[
\begin{aligned}
\text{LV} &= -0.500000163912773 \quad -0.500000193715096 \\
\text{H} &= \quad 6.00000035762787 \quad 2.00000041723251
\end{aligned}
\]

Total number of zeros inside this box = 6

Number of boxes containing at most 5 zeros = 2

These boxes are given by:

1)  \[
\begin{aligned}
\text{LV} &= -0.500000163912773 \quad -0.500000193715096 \\
\text{H} &= \quad 3.00000017881393 \quad 2.00000041723251
\end{aligned}
\]
Total number of zeros inside this box = 4

2)  \[
\begin{aligned}
\text{LV} &= \quad 2.50000001490116 \quad -0.500000193715096 \\
\text{H} &= \quad 3.00000017881393 \quad 2.00000041723251
\end{aligned}
\]
Total number of zeros inside this box = 2

Final approximations for the zeros and verification:

1)  Number of mutually distinct zeros = 3

\[
\begin{aligned}
z &= (-0.444089209850063\_E-15, -0.284397851396988\_E-16) \\
f(z) &= (0.491016012316152\_E-29, 0.631490085549722\_E-30)
\end{aligned}
\]
\[ \text{multiplicity} = 2 \]

\[ z = (1.1890658973011, 0.840925724965599E-27) \]

\[ f(z) = (0.000000000000000, -0.332912397884017E-26) \]

\[ \text{multiplicity} = 1 \]

\[ z = (1.72843498616506, -0.366031460022549E-27) \]

\[ f(z) = (-0.222044604925031E-15, -0.174867254641548E-26) \]

\[ \text{multiplicity} = 1 \]

2) **Number of mutually distinct zeros** = 2

\[ z = (4.03038191606047, 0.288920306537196E-28) \]

\[ f(z) = (0.155431223447522E-13, 0.308650229769291E-26) \]

\[ \text{multiplicity} = 1 \]

\[ z = (3.01990732809571, 0.185382312726938E-28) \]

\[ f(z) = (-0.105471187339390E-14, -0.402275402644635E-27) \]

\[ \text{multiplicity} = 1 \]

**Example 10.** Finally, suppose that \( f(z) = z^2(z - 2)^2[e^{2z} \cos z + z^3 - 1 - \sin z] \)

and let \( W \) be the region determined by

\[ LV = (-1.0_{DP}, -1.0_{DP}) \text{ and } H = (4.0_{DP}, 2.0_{DP}). \]

Note that \( f \) has a triple zero at the origin and a double zero at \( z = 2 \). We set \( M = 5 \)

and \( ICON = 3 \).

This is ZEAL. Version of October 1998.

Input:

\[ LV = -1.000000000000000 \quad -1.000000000000000 \]

\[ H = 4.000000000000000 \quad 2.000000000000000 \]

\[ M = 5 \]

\[ ICON = 3 \]

FILES = T

Results:

The following box has been considered:

\[ LV = -1.00000016391277 \quad -1.00000019371510 \]

\[ H = 4.00000035762787 \quad 2.00000041723251 \]

Total number of zeros inside this box = 8
Number of boxes containing at most 5 zeros = 2

These boxes are given by:

1) \[ \begin{align*}
L &= -1.00000016391277 & -1.00000019371510 \\
H &= 2.00000017881393 & 2.00000041723251 \\
\text{Total number of zeros inside this box} &= 5
\end{align*} \]

2) \[ \begin{align*}
L &= 1.00000001490116 & -1.00000019371510 \\
H &= 2.00000001788139 & 2.00000041723251 \\
\text{Total number of zeros inside this box} &= 3
\end{align*} \]

Final approximations for the zeros and verification:

1) Number of mutually distinct zeros = 3

\[ \begin{align*}
z &= (-0.460714119728971, 0.625427769347768) \\
f(z) &= (-0.125408855493498E-14, -0.242634956938915E-14) \\
multiplicity &= 1
\end{align*} \]

\[ \begin{align*}
z &= (-0.555111512312578E-16, 0.780756160460674E-15) \\
f(z) &= (0.270707981407843E-45, -0.189411036718915E-44) \\
multiplicity &= 3
\end{align*} \]

\[ \begin{align*}
z &= (-0.460714119728972, -0.625427769347767) \\
f(z) &= (0.420110659916658E-14, 0.555338820267851E-14) \\
multiplicity &= 1
\end{align*} \]

2) Number of mutually distinct zeros = 2

\[ \begin{align*}
z &= (2.00000000000000, 0.105626554996077E-14) \\
f(z) &= (-0.253754972900138E-27, -0.312032161709746E-27) \\
multiplicity &= 2
\end{align*} \]

\[ \begin{align*}
z &= (1.66468286974552, -0.528536806498078E-28) \\
f(z) &= (0.2767417773088933E-15, 0.405540193714869E-27) \\
multiplicity &= 1
\end{align*} \]

5.5. Concluding remarks. We have applied our package to various analytic functions and rectangular regions and we have found that it behaves predictably and accurately. ZEAL calculates the total number of zeros that lie inside the given box and then computes approximations for these zeros, together with their respective multiplicities. Our package does not require initial approximations for the zeros.

The user will appreciate the flexibility offered by the input parameter ICON. If nothing is known about the zeros that lie inside the given box, one may call ZEAL with ICON = 1 to obtain the total number of zeros. Then one may proceed with ICON = 3 to compute approximations for all these zeros, or, if less than the total
number of zeros are required, with $\text{ICON} = 4$ and $\text{NR}$ equal to the requested number of zeros. If only a set of boxes is required, each of which contains less than $M$ zeros (counting multiplicities), then one may set $\text{ICON} = 2$.

6. A derivative-free approach

The results presented in the previous sections are based on the form (23), which involves the logarithmic derivative $f'/f$. Instead, let us consider the symmetric bilinear form

$$\langle \cdot , \cdot \rangle_\ast : \mathcal{P} \times \mathcal{P} \to \mathbb{C}$$

defined as

$$\langle \phi, \psi \rangle_\ast := \frac{1}{2\pi i} \int_\gamma \phi(z)\overline{\psi(z)}\frac{1}{f(z)} \, dz$$

for any two polynomials $\phi, \psi \in \mathcal{P}$. Again, this form can be evaluated via numerical integration along $\gamma$ and in what follows we will assume that all the “inner products” $\langle \phi, \psi \rangle_\ast$ that are needed have been calculated.

We will show that essentially the same results can be obtained with the form $\langle \cdot , \cdot \rangle_\ast$ as previously with $\langle \cdot , \cdot \rangle$. Note that the derivative $f'$ is no longer needed. Of course, in this new approach not the mutually distinct zeros but rather the unknowns $Z_1, \ldots, Z_N$ (introduced by Delves and Lyness, cf. Section 1) are calculated and the multiplicities cannot be computed explicitly. But apart from this, the approach has the same advantages as the algorithm (which we will henceforth call ‘our Algorithm’) that we have presented in Section 3. In particular, it is self-starting in the sense that it does not require initial approximations for the zeros.

As by assumption the derivative $f'$ is not available to us, we cannot obtain the value of $N$ by evaluating the integral in the right-hand side of (14) numerically. Instead one can use the principle of the argument, cf. our discussion in Subsection 1.1.

The integrand that appears in the right-hand side of (37) has a pole at every zero of $f$ that lies in the interior of $\gamma$ and the order of the pole is equal to the multiplicity of the zero. Therefore, the residue theorem implies that $\langle \phi, \psi \rangle_\ast$ is equal to the sum of the residues of the function $\phi\psi/f$ at these poles. The following result can easily be verified.

**Proposition 11.** Suppose that all the $N$ zeros $Z_1, \ldots, Z_N$ of $f$ that lie inside $\gamma$ are simple. Then

$$\langle \phi, \psi \rangle_\ast = \sum_{k=1}^{N} \frac{\phi(Z_k)\psi(Z_k)}{f'(Z_k)}.$$

In general, if $f$ has multiple zeros, then an elegant expression for $\langle \phi, \psi \rangle_\ast$ written as a sum is much more difficult to obtain. Fortunately, it is not necessary to have such an expression available. The proofs of Theorems 3 and 4 depend completely on the details of the way in which $\langle \phi, \psi \rangle$ can be written as a sum, cf. Equation (23). However, as we will see, the corresponding theorems can also be proved in a different way.
Define $s_p^* := (1, z^p)$ for $p = 0, 1, 2, \ldots$ and let $H_k^*$ be the $k \times k$ Hankel matrix

$$H_k^* := \left[ s_{p+q}^* \right]_{p, q=0}^{k-1}$$

for $k = 1, 2, \ldots$. The formal orthogonal polynomials associated with $\langle \cdot, \cdot \rangle_*$ can be defined as before. The coefficients of regular FOPs can be computed by solving a Yule-Walker system, cf. Equation (25). Also, $t \geq 1$ is a regular index if and only if the matrix $H_t^*$ is nonsingular.

The residue theorem immediately implies that the polynomial

$$P_N(z) = \prod_{k=1}^N (z - Z_k)$$

satisfies

$$(z^p, P_N(z))_* = 0, \quad p = 0, 1, 2, \ldots$$

In this sense, the polynomial $P_N(z)$ behaves with respect to the form $\langle \cdot, \cdot \rangle_*$ in the same way as the polynomial $\varphi_n(z)$ behaves with respect to $\langle \cdot, \cdot \rangle$, cf. Equations (27) and (28). We will prove that $N$ is the largest regular index for $\langle \cdot, \cdot \rangle_*$. This will enable us to compute the zeros of the regular FOP $P_N(z)$, i.e., the zeros $Z_1, \ldots, Z_N$, in essentially the same way as our Algorithm applied to the form $\langle \cdot, \cdots \cdot \rangle$ computes the zeros of $\varphi_n(z)$, i.e., the mutually distinct zeros $z_1, \ldots, z_n$.

The following lemma will play an important role. Define the set $\mathcal{I}$ as follows:

$$\mathcal{I} := \{ \phi \in \mathcal{P} : \langle z^p, \phi(z) \rangle_* = 0 \quad \text{for} \quad p = 0, 1, 2, \ldots \}. $$

**Lemma 12.** The set $\mathcal{I}$ is equal to the ideal generated by the polynomial $P_N$. In other words,

$$\mathcal{I} = \{ \phi \in \mathcal{P} : \exists \alpha \in \mathcal{P} : \phi = \alpha P_N \}.$$

**Proof.** Suppose that $a \in \mathbb{C}$ lies in the interior of $\gamma$. Let the function $g : W \to \mathbb{C}$ be meromorphic and suppose that $g$ has neither zeros nor poles on $\gamma$. Then the coefficient of $(z - a)^{p-1}$ in the Laurent expansion of $g$ at the point $a$ is given by the integral

$$\frac{1}{2\pi i} \int_\gamma (z - a)^p g(z) \, dz$$

for $p = 0, 1, 2, \ldots$. Let $\phi \in \mathcal{I}$. Then

$$\langle (z - Z_k)^p, \phi(z) \rangle_* = \frac{1}{2\pi i} \int_\gamma (z - Z_k)^p \frac{\phi(z)}{f(z)} \, dz = 0$$

for $k = 1, \ldots, N$ and $p = 0, 1, 2, \ldots$ and thus the function $\frac{\phi}{f}$ has a removable singularity at the points $Z_1, \ldots, Z_N$. Thus $\phi$ has to be a multiple of $P_N$. This proves the lemma.

**Theorem 13.** The matrix $H_N^*$ is nonsingular.

**Proof.** We will prove that $P_N$ is the only monic polynomial of degree $N$ that is orthogonal to all polynomials of lower degree. Suppose that $Q_N$ is another such polynomial. Then $P_N - Q_N$ is of degree at most $N-1$ and hence $\langle P_N - Q_N, Q_N \rangle_*$ $= 0$. Equation (38) then implies that $\langle Q_N, Q_N \rangle_* = 0$. Thus $Q_N$ is not only orthogonal
to all polynomials of degree \( \leq N - 1 \) but also to all polynomials of degree \( N \). The polynomial \( z^N - Q_N \) has degree \( \leq N \) and therefore \( \langle z^N - Q_N, Q_N \rangle_s = 0 \). As \( \langle z^N, Q_N \rangle_s = \langle P_N, z^N \rangle_s = 0 \), it follows that \( \langle z_Q, Q_N \rangle_s = 0 \). Thus \( Q_N \) is also orthogonal to all polynomials of degree \( N + 1 \). By continuing this way, one can prove that \( Q_N \) is orthogonal to all polynomials, \( Q_N \in I \). As \( Q_N \) is a monic polynomial of degree \( N \), the previous lemma then implies that \( Q_N = P_N \). Thus there is only one monic polynomial of degree \( N \) that is orthogonal to all polynomials of lower degree. This implies that the matrix \( H_N^* \) is nonsingular.

**Theorem 14.** The matrix \( H_{N+k}^* \) is singular for \( k = 1, 2, \ldots \).

**Proof.** Instead of the basis of the monomials \( \{ z^p \}_{p \geq 0} \) we consider the basis \( \{ \psi_p(z) \}_{p \geq 0} \) where \( \psi_p(z) := z^p \) for \( p = 0, 1, \ldots, N - 1 \) and \( \psi_N + p(z) := z^p P_N(z) \) for \( p = 0, 1, 2, \ldots \). Let

\[
F_i^* := \left[ \langle \psi_p, \psi_q \rangle_s \right]_{p,q=0}^{l-1}
\]

be the corresponding \( l \times l \) Gram matrix for \( l = 1, 2, \ldots \). Equation (38) then implies that \( \det F_{N+k}^* = 0 \) for \( k = 1, 2, \ldots \). One can easily verify that \( \det F_i^* = \det H_i^* \) for \( l = 1, 2, \ldots \). This proves the theorem. \( \square \)

We have now identified \( P_N(z) \) as the regular FOP of degree \( N \) and we have shown that regular FOPs of degree larger than \( N \) do not exist. Note that \( s_0^* \) is equal to the sum of the residues of \( 1/f \) at the points \( \{ Z_1, \ldots, Z_N \} \) and hence it is not necessarily different from zero. Therefore, the regular FOP of degree \( N \) with respect to the form \( \langle \cdot, \cdot \rangle_s \) doesn’t always exist, in contrast to \( \langle \cdot, \cdot \rangle \).

The zero/eigenvalue properties discussed in Section 3 hold not only for \( \langle \cdot, \cdot \rangle \) but for every symmetric bilinear form. The zeros \( Z_1, \ldots, Z_N \) can therefore also be calculated by solving a generalized eigenvalue problem. The following result can be proved in the same way as Theorem 4. Let \( H_k^{*<} \) be the Hankel matrix

\[
H_k^{*<} := \begin{bmatrix}
    s_1^* & s_2^* & \cdots & s_k^* \\
    s_2^* & \ddots & \vdots & \vdots \\
    \vdots & \ddots & \ddots & \vdots \\
    s_k^* & \cdots & \cdots & s_{2k-1}^*
\end{bmatrix}
\]

for \( k = 1, 2, \ldots \).

**Theorem 15.** The eigenvalues of the pencil \( H_N^{*<} - \lambda H_N^* \) are given by \( Z_1, \ldots, Z_N \).

Below we will compare the accuracy obtained via Theorem 4 to the accuracy obtained via Theorem 15.

The zeros \( Z_1, \ldots, Z_N \) can also be computed by applying our Algorithm to the form \( \langle \cdot, \cdot \rangle_s \). Let \( \{ \varphi_r^* \}_{r \geq 0} \) denote the FOPs associated with \( \langle \cdot, \cdot \rangle_s \). Define the matrices \( G_k^r \) and \( G_k^{*z} \) as

\[
G_k^r := \left[ \langle \varphi_r^*, \varphi_s^* \rangle \right]_{r,s=0}^{k-1} \quad \text{and} \quad G_k^{*z} := \left[ \langle \varphi_r^*, z \varphi_s^* \rangle \right]_{r,s=0}^{k-1}
\]

for \( k = 1, 2, \ldots \). The following results can be proved in the same way as Theorem 7 and Corollary 8.
Theorem 16. Let $t \geq 1$ be a regular index for $\langle \cdot, \cdot \rangle_p$ and let $z_{1,t}^*, \ldots, z_{t,t}^*$ be the zeros of the regular FOP $\varphi_p^*$. Then the eigenvalues of the pencil $G_p^* - \lambda G_{p,t}$ are given by $z_{1,t}^*, \ldots, z_{t,t}^*$.

Corollary 17. The eigenvalues of $G_p^* - \lambda G_{p,t}$ are given by $Z_1, \ldots, Z_N$.

If instead of $N$ only an upper bound for $N$ is available, then the value of $N$ can be computed via the stopping criterion of our Algorithm.

We will now discuss a few numerical examples. The computations have been done via Matlab 5 (with floating point relative accuracy $\approx 2.2204 \times 10^{-16}$). The integration algorithm is the same as the one discussed at the beginning of Section 4.

Example 11. Let $f(z) = e^{3z} - 2z \cos z - 1$. Suppose that $\gamma$ is the circle $\gamma = \{ z \in \mathbb{C} : |z| = 4 \}$. Then $N = 6$. Let us try an approach based on ordinary moments. Table 1 contains approximations for $s_p = \langle 1, z^p \rangle$ and $s_p^* = \langle 1, z^p \rangle^*$ for $p = 0, 1, \ldots, 11$. Note that in both cases the order of magnitude changes as $p$ increases. The computed approximations for the zeros $Z_1, \ldots, Z_N$ obtained via

\[
\begin{array}{c|cc}
p & s_p & s_p^* \\
\hline
0 & 6.0 & -7.510^{-2} \\
1 & 2.0 & 2.810^{-1} \\
2 & -1.410^1 & -9.110^{-1} \\
3 & -8.510^1 & 2.1 \\
4 & -3.010^1 & -2.0 \\
5 & 7.010^2 & 2.4 \\
6 & 2.510^3 & -2.310^1 \\
7 & -1.010^3 & 1.5101 \\
8 & -3.110^4 & 9.910^4 \\
9 & -7.610^4 & 4.610^2 \\
10 & 1.310^5 & -4.810^2 \\
11 & 1.210^6 & -5.310^3 \\
\end{array}
\]

Table 1. Ordinary moments $s_p$ and $s_p^*$

Theorem 4 and 15 are shown in Table 2 and 3, respectively. The digits that are not correct are underlined. Observe that the approximations for the zeros are very

\[-2.186079491175828 \times 10^{-13} - i 1.72762308312215310^{-12} \\
5.308949329942010^{-1} + i 1.331791876750615 \\
5.30894932992837610^{-1} - i 1.331791876751221 \\
-1.844233933262199 - i 1.20449415204231710^{-14} \\
1.414607177658159 + i 3.047722062627169 \\
1.414607177658159 - i 3.047722062627173 \\
\]

Table 2. Approximations for the zeros obtained via the ordinary moments $s_p$. 

accurate. Using ordinary moments has the advantage that only $2N$ integrals have to be calculated and hence, compared to our Algorithm, the arithmetic cost is rather limited. Also, a significant part of the computation required for each integrand is the same for all of the integrands (namely, the computation of $f'/f$ or $1/f$). By programming the quadrature algorithm in such a way that it is able to integrate a vector of similar integrals, these common calculations need be done only once for each integrand evaluation point. However, as the following example shows, ordinary moments do not always lead to such accurate results.

**Example 12.** The Wilkinson polynomial and also functions that have clusters of zeros are typical, although somewhat extreme, examples where an approach based on ordinary moments is likely to fail. The following function is another example. Suppose that $f(z) = J_0(z)$, the Bessel function of the first kind and of order zero. It is known that this function has only positive real zeros and that all these zeros are simple (see, e.g., Watson [295] and also Chapter 5). In that sense it is related to the Wilkinson polynomial. Suppose that $\gamma = \{ z \in \mathbb{C} : |z - 15| = 14.5 \}$. Then $N = 9$.

<table>
<thead>
<tr>
<th>exact zeros</th>
<th>$s_p^*$</th>
<th>$\langle \cdot , \cdot \rangle_*$</th>
<th>$s_p$</th>
<th>$\langle \cdot , \cdot \rangle$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4048255557695773</td>
<td>5</td>
<td>12</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>5.520078110286311</td>
<td>2</td>
<td>11</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>8.653727912911013</td>
<td>2</td>
<td>10</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>11.79153443901428</td>
<td>3</td>
<td>11</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>14.93091776848778</td>
<td>3</td>
<td>11</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>18.07106396791092</td>
<td>3</td>
<td>11</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>21.21163662987936</td>
<td>4</td>
<td>11</td>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>24.35247153074930</td>
<td>5</td>
<td>11</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>27.49347913204025</td>
<td>7</td>
<td>11</td>
<td>7</td>
<td>12</td>
</tr>
</tbody>
</table>

**Table 4.** The number of correct significant digits in case $f(z) = J_0(z)$. Observe that in both cases the approximations obtained via our Algorithm are more accurate than the approximations obtained via ordinary moments. Of course, there is clearly a trade-off between obtained accuracy and cost. We advise the reader to start
with the cheapest approach, i.e., the approach based on the ordinary moments \( s_p \). If the computed approximations for the zeros are not sufficiently accurate to be refined via an iterative method (one that doesn’t need the derivative, of course), then one can apply our Algorithm to the form \( \langle \cdot, \cdot \rangle \), or switch to one of the approaches that use both \( f \) and \( f' \).

**Example 13.** Let us illustrate how the stopping criterion of our Algorithm can be used to determine the value of \( N \) in case only an upper bound for \( N \) is known.

Consider again the function \( f(z) = e^{\beta z} - 2z \cos z - 1 \) and suppose that \( \gamma \) is the circle \( \gamma = \{ z \in \mathbb{C} : |z| = 5 \} \). Then \( N = 7 \). Let us assume that only the upper bound 20 is known. Our algorithm defines the FOP \( \varphi^*_1 \) as an inner polynomial and \( \varphi^*_2 \) as a regular FOP. At this point the algorithm asks itself whether \( N \) is equal to two. It computes \( |\langle \varphi^*_2, \varphi^*_2 \rangle| \). To take into account the accuracy lost during the evaluation of the quadrature formula, this quantity is scaled in a certain way, cf. Section 3. The resulting floating point number is given by

\[
1.998543018990362,
\]

which is certainly not “sufficiently small” (we use \( 10^{-8} \) as a threshold) and hence the algorithm continues. It defines \( \varphi^*_3 \) as an inner polynomial and \( \varphi^*_4 \) as a regular FOP. Then it checks if \( N \) is equal to four. It compares

\[
1.981687381683116
\]

to \( 10^{-8} \) and continues. The polynomial \( \varphi^*_4 \) is defined as a regular FOP. The algorithm again decides to continue and defines \( \varphi^*_5 \) as a regular FOP. The corresponding floating point number is given by

\[
0.3794464188056766
\]

and the algorithm continues. It defines \( \varphi^*_6 \) as a regular FOP. We have now reached the actual value of \( N \). The scaled counterparts of the inner products that correspond to the sequence (34) are given by

\[
\begin{align*}
9.190814944765118 \times 10^{-16} & \\
1.799485800789563 \times 10^{-15} & \\
4.008700446099430 \times 10^{-15} & \\
5.548436809880727 \times 10^{-15} & \\
6.603511781861131 \times 10^{-15} & \\
5.538342691314587 \times 10^{-15} & \\
3.496340876196310 \times 10^{-15} & \\
4.116380174988637 \times 10^{-16} & \\
5.379567405602837 \times 10^{-15} & \\
8.806423950940129 \times 10^{-15} & \\
8.912210606016121 \times 10^{-15} & \\
5.866528582137192 \times 10^{-15} & \\
1.127215921207880 \times 10^{-15} & 
\end{align*}
\]

and hence the algorithm decides that \( N \) is equal to seven and it stops. The computed approximations for the zeros are given by
Then one can easily show that the generalized eigenvalue problem are given by the zeros basis \( \text{diagonal entries are different from zero because of roundoff errors and Matlab returns their quotient as an eigenvalue. Thus, by solving the } 20 \times 20 \text{ generalized eigenvalue problem } H_{20}^* = \lambda H_{20} \text{ we obtain approximations for the seven zeros } Z_1, \ldots, Z_N \text{ and 13 spurious eigenvalues. The latter can be detected by evaluating } f \text{ at the computed eigenvalues and also by taking into account that the computed approximations for the zeros are likely to lie inside } \gamma \text{ or at least quite close to it. The approximations for the zeros obtained in this way are 1 to 3 digits less accurate than the approximations obtained via our Algorithm. By solving the } 7 \times 7 \text{ generalized eigenvalue problem, one obtains approximations that are about as accurate as those computed by our Algorithm.}

**Example 14.** Let us consider a function that has multiple zeros. Suppose that

\[
f(z) = z^2(z - 2)^2 \left[ e^{2z} \cos z + z^3 - 1 - \sin z \right]
\]

and let \( \gamma = \{ z \in \mathbb{C} : |z| = 3 \} \). Note that \( f \) has a triple zero at the origin and a double zero at \( z = 2 \). The total number of zeros of \( f \) that lie inside \( \gamma \) is equal to eight, \( N = 8 \). By using the ordinary moments \( s_n \) we obtain the following approximations for the zeros.
We have underlined the correct significant digits. Our Algorithm gives comparable results. Note how the obtained accuracy diminishes as the multiplicity of the zero increases.

### Acknowledgements

The results in Section 3 and 6 were obtained in collaboration with Marc Van Barel.

Norman Katz sent me the pages of [306] that are related to Theorem 2 whereas Nikolaos Ioakimidis gave me reprints of more than 25 papers of his. I would also like to thank James Lyness and Tien-Yien Li for stimulating conversations and Koen Engelborghs for giving me the permission to include Figure 3.

The first version of the package ZEAL was written during my second research stay (March 22–April 18, 1998) at the Department of Mathematics, University of Patras (Patras, Greece). I would like to thank Michael Vrahatis, Omiros Ragos and Filareti Zafiropoulous for their very kind hospitality and for giving me the opportunity to collaborate with them. I would also like to thank Vlaamse Leergangen Leuven for its generous financial support.

I thank Bernard Mourrain for the stimulating discussions that initiated the results presented in Section 6.

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$1.18353131559952610^{-4}$</td>
<td>$-5.99490479430230010^{-5}$</td>
<td>$-5.84121833536418410^{-5}$</td>
<td>$2.000000113260292$</td>
</tr>
<tr>
<td></td>
<td>$+i 8.8406481378441010^{-7}$</td>
<td>$+i 1.02052244544141410^{-4}$</td>
<td>$+i 1.02936309346076810^{-4}$</td>
<td>$+i 9.25372740200930610^{-7}$</td>
</tr>
<tr>
<td></td>
<td>$1.99999886743732$</td>
<td>$-9.25373678838278510^{-7}$</td>
<td>$-4.60714119728599510^{-1}$</td>
<td>$+6.25427769347138010^{-1}$</td>
</tr>
<tr>
<td></td>
<td>$+i 6.25427769347288110^{-1}$</td>
<td>$+i 6.25427769347288110^{-1}$</td>
<td>$-4.60714119728724610^{-1}$</td>
<td>$-4.60714119728724610^{-1}$</td>
</tr>
<tr>
<td></td>
<td>$1.664682869740608$</td>
<td>$+i 1.09330745522126510^{-12}$</td>
<td>$1.664682869740608$</td>
<td>$1.664682869740608$</td>
</tr>
</tbody>
</table>

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Clusters of zeros of analytic functions

In the previous chapter we have presented an accurate algorithm, based on the theory of formal orthogonal polynomials, for computing zeros of analytic functions. More specifically, given an analytic function \( f \) and a Jordan curve \( \gamma \) that does not pass through any zero of \( f \), we have considered the problem of computing all the zeros \( z_1, \ldots, z_n \) of \( f \) that lie inside \( \gamma \), together with their respective multiplicities \( \nu_1, \ldots, \nu_n \). Our principal means of obtaining information about the location of these zeros has been the symmetric bilinear form \( \langle \cdot, \cdot \rangle \), cf. Equation (23). This form can be evaluated via numerical integration along \( \gamma \).

This chapter continues the previous chapter. If \( f \) has one or several clusters of zeros, then the mapping from the ordinary moments associated with \( \langle \cdot, \cdot \rangle \) to the zeros and their respective multiplicities is very ill-conditioned. We will show that the algorithm that we have presented in Chapter 1 can be used to calculate the centre of a cluster and its size, i.e., the arithmetic mean of the zeros that form a certain cluster and the total number of zeros in this cluster, respectively. This information enables one to zoom into a certain cluster: its zeros can be calculated separately from the other zeros of \( f \). By shifting the origin in the complex plane to the centre of a certain cluster, its zeros become better relatively separated, which is appropriate in floating point arithmetic and reduces the ill-conditioning.

In this chapter we will also attack our problem of computing all the zeros of \( f \) that lie inside \( \gamma \) in an entirely different way, based on rational interpolation at roots of unity. We will show how the new approach complements the previous one and how it can be used effectively in case \( \gamma \) is the unit circle.

Note. Specifically for clusters of polynomial zeros, let us mention that Hribernik and Stetter [149] worked on detection and validation of clusters of zeros whereas Kirrinnis [178] studied Newton’s iteration towards a cluster.

This chapter corresponds to part of our paper [186].

1. How to obtain the centre of a cluster and its weight

Suppose that the zeros of \( f \) that lie inside \( \gamma \) can be grouped into \( m \) clusters. Let \( I_1, \ldots, I_m \) be index sets that define these clusters, and let

\[
\mu_j := \sum_{k \in I_j} \nu_k \quad \text{and} \quad c_j := \frac{1}{\mu_j} \sum_{k \in I_j} \nu_k z_k
\]

for \( j = 1, \ldots, m \). In other words, \( \mu_j \) is equal to the total number of zeros that form cluster \( j \) (its “weight”) whereas \( c_j \) is equal to the arithmetic mean of the zeros
in cluster \( j \) (its “centre of gravity”). We assume that the centres \( c_1, \ldots, c_m \) are mutually distinct. For \( k = 1, \ldots, n \) we also define \( \zeta_k := z_k - e_j \) if \( k \in I_j \). From the definition of \( \mu_j \) and \( c_j \) it follows that

$$
\sum_{k \in I_j} \nu_k \zeta_k = 0, \quad j = 1, \ldots, m.
$$

Define the symmetric bilinear form \( \langle \cdot, \cdot \rangle_m \) by

$$
\langle \phi, \psi \rangle_m := \sum_{j=1}^m \mu_j \phi(c_j) \psi(c_j)
$$

for any two polynomials \( \phi, \psi \in \mathcal{P} \). This form is related to the form \( \langle \cdot, \cdot \rangle \) in an obvious way: instead of the zeros \( z_1, \ldots, z_n \) and their multiplicities \( \nu_1, \ldots, \nu_n \), we now use the centres of gravity \( c_1, \ldots, c_m \) and the weights \( \mu_1, \ldots, \mu_m \) of the clusters. Let

$$
\delta := \max_{1 \leq k \leq n} |\zeta_k|.
$$

The following theorem tells us that \( \langle \cdot, \cdot \rangle_m \) approximates \( \langle \cdot, \cdot \rangle \) (and vice versa).

**Theorem 19.** Let \( \phi, \psi \in \mathcal{P} \). Then \( \langle \phi, \psi \rangle = \langle \phi, \psi \rangle_m + \mathcal{O}(\delta^2) \), \( \delta \rightarrow 0 \).

**Proof.** The following holds:

$$
\langle \phi, \psi \rangle = \sum_{k=1}^n \nu_k \phi(z_k) \psi(z_k)
$$

$$
= \sum_{j=1}^m \sum_{k \in I_j} \nu_k \phi(c_j + \zeta_k) \psi(c_j + \zeta_k)
$$

$$
= \sum_{j=1}^m \sum_{k \in I_j} \nu_k \left( \phi(c_j) \psi(c_j) + \zeta_k [\phi(z) \psi(z)]'_{z=c_j} + \mathcal{O}(\zeta_k^2), \quad \zeta_k \rightarrow 0 \right)
$$

$$
= \sum_{j=1}^m \mu_j \phi(c_j) \psi(c_j) + \sum_{j=1}^m \sum_{k \in I_j} \nu_k \zeta_k \left[\phi(z) \psi(z)\right]'_{z=c_j} + \sum_{k=1}^n \mathcal{O}(\zeta_k^2), \quad \zeta_k \rightarrow 0
$$

$$
= \sum_{j=1}^m \mu_j \phi(c_j) \psi(c_j) + \sum_{k=1}^n \mathcal{O}(\zeta_k^2), \quad \zeta_k \rightarrow 0.
$$

This proves the theorem. \( \square \)

Define the ordinary moments associated with \( \langle \cdot, \cdot \rangle_m \) as

$$
s_p^{(m)} := \langle 1, z^p \rangle_m
$$
for \( p = 0, 1, 2, \ldots \). Observe that \( s_0^{(m)} = s_0 \) whereas \( s_1^{(m)} = s_1 \). Define the vectors \( s, s^{(m)} \in \mathbb{C}^{N-1} \) as

\[
\mathbf{s} := \begin{bmatrix}
    s_0 \\
    s_1 \\
    \vdots \\
    s_{2N-2}
\end{bmatrix}
\quad \text{and} \quad
\mathbf{s}^{(m)} := \begin{bmatrix}
    s_0^{(m)} \\
    s_1^{(m)} \\
    \vdots \\
    s_{2N-2}^{(m)}
\end{bmatrix}.
\]

The entries of \( s \) determine the Hankel matrix \( H_N \). The previous theorem implies that

\[
\frac{||s - s^{(m)}||_2}{||s||_2} = O(\delta^2), \ \delta \to 0.
\]

Let \( H_k^{(m)} \) be the \( k \times k \) Hankel matrix

\[
H_k^{(m)} := \begin{bmatrix}
    s_0^{(m)} & s_1^{(m)} & \cdots & s_{k-1}^{(m)} \\
    s_1^{(m)} & s_0^{(m)} & \cdots & s_{k-2}^{(m)} \\
    \vdots & \vdots & \ddots & \vdots \\
    s_{k-1}^{(m)} & s_{k-2}^{(m)} & \cdots & s_0^{(m)}
\end{bmatrix}_{p,q=0}^{k-1}
\]

for \( k = 1, 2, \ldots \).

**Corollary 20.** Let \( k \geq 1 \). Then \( \det H_k = \det H_k^{(m)} + O(\delta^2), \ \delta \to 0. \)

**Proof.** Let \( k \) be a positive integer. Then the previous theorem implies that

\[
H_k = \begin{bmatrix}
    s_0^{(m)} & s_1^{(m)} & \cdots & s_{k-1}^{(m)} \\
    s_1^{(m)} & s_0^{(m)} & \cdots & s_{k-2}^{(m)} \\
    \vdots & \vdots & \ddots & \vdots \\
    s_{k-1}^{(m)} & s_{k-2}^{(m)} & \cdots & s_0^{(m)}
\end{bmatrix}_{p,q=0}^{k-1}.
\]

The result follows by expanding the determinant of the matrix in the right-hand side.

**Corollary 21.** The matrix \( H_m \) is nonsingular if \( \delta \to 0 \). Let \( t > m \). Then \( \det H_t = O(\delta^t), \ \delta \to 0. \)

**Proof.** This follows from the previous corollary and the fact that \( H_m^{(m)} \) is nonsingular whereas \( H_t^{(m)} \) is singular for all integers \( t > m \) (cf. Theorem 3).

The following theorem should be compared with Theorem 9.

**Theorem 22.** Let \( t \) be an integer \( \geq m \). Then \( \varphi_t(c_j) = O(\delta^t), \ \delta \to 0 \) for \( j = 1, \ldots, m \). Also \( \langle z^p, \varphi_t(z) \rangle = O(\delta^t), \ \delta \to 0 \) for all \( p \geq t \).

**Proof.** Let \( t \geq m \). If \( t \) is a regular index, then

\[
\langle z^p, \varphi_t(z) \rangle = 0, \quad p = 0, 1, \ldots, t - 1,
\]

else

\[
\langle z^p, \varphi_t(z) \rangle = 0, \quad p = 0, 1, \ldots, r - 1
\]

where \( r \) is the largest regular index less than \( t \). Corollary 21 implies that \( r \geq m \), and thus we may conclude that

\[
\langle z^p, \varphi_t(z) \rangle = 0, \quad p = 0, 1, \ldots, m - 1.
\]

Theorem 19 then implies that

\[
\langle z^p, \varphi_t(z) \rangle_m = O(\delta^t), \ \delta \to 0, \quad p = 0, 1, \ldots, m - 1.
\]
In matrix notation this can be written as

\[
\begin{bmatrix}
1 & \cdots & 1 \\
c_1 & \cdots & c_m \\
\vdots & \vdots & \vdots \\
c_{m-1} & \cdots & c_{m-1}
\end{bmatrix}
\begin{bmatrix}
\mu_1 \\
\mu_2 \\
\vdots \\
\mu_m
\end{bmatrix}
= \begin{bmatrix}
\varphi_1(c_1) \\
\varphi_1(c_2) \\
\vdots \\
\varphi_1(c_m)
\end{bmatrix} = O(\delta^2), \; \delta \to 0.
\]

The right-hand side represents a vector in \( \mathbb{C}^m \) whose entries are \( O(\delta^2) \), \( \delta \to 0 \). As the centres \( c_1, \ldots, c_m \) are assumed to be mutually distinct and the weights \( \mu_1, \ldots, \mu_m \) are different from zero, it follows that

\[
\varphi_1(c_j) = O(\delta^2), \; \delta \to 0, \quad j = 1, \ldots, m.
\]

Theorem 19 then immediately implies that

\[
\langle z^p, \varphi_1(z) \rangle = O(\delta^2), \; \delta \to 0
\]

for all \( p \geq t \).

In other words, unless the FOP \( \varphi_1(z) \) has a very flat shape near its zeros, we are likely to find good approximations for the centres \( c_1, \ldots, c_m \) among the zeros of \( \varphi_1(z) \) for all \( t \geq m \). Note that

\[
\begin{bmatrix}
1 & \cdots & 1 \\
c_1 & \cdots & c_m \\
\vdots & \vdots & \vdots \\
c_{m-1} & \cdots & c_{m-1}
\end{bmatrix}
\begin{bmatrix}
\mu_1 \\
\mu_2 \\
\vdots \\
\mu_m
\end{bmatrix}
= \begin{bmatrix}
s_0 \\
s_1 \\
\vdots \\
s_{m-1}
\end{bmatrix} + O(\delta^2), \; \delta \to 0.
\]

It follows that approximations for the weights \( \mu_1, \ldots, \mu_m \) can be obtained by solving a Vandermonde system.

What happens if we apply the algorithm that we have presented in Chapter 1 in case the zeros of \( f \) can be grouped into clusters? The second part of Theorem 22 implies that our algorithm stops at \( r = m \) if \( \delta \), the maximal size of the clusters, is sufficiently small. It returns the zeros of the FOP \( \varphi_m(z) \) that is associated with \( \langle \cdot, \cdot \rangle \). Theorem 19 and the fact that the \( m \)th degree FOP with respect to \( \langle \cdot, \cdot \rangle_m \) is given by \( \prod_{j=1}^m (z - c_j) \) imply that we can use these zeros as approximations for the centres of the clusters. (This also follows from the first part of Theorem 22, of course.) The computed approximations for the weights of the clusters should be close to integers.

We can check this to verify that we have indeed determined the correct value of \( m \). We can also calculate (approximations for) the ordinary moments asso ciated with \( \langle \cdot, \cdot \rangle_m \) and verify if (39) is satisfied.

2. A numerical example

In the following example we have again considered the case that \( \gamma \) is a circle, cf. Section 4 of Chapter 1. The computations have been done via Matlab 5.

Stewart’s perturbation theory for the generalized eigenvalue problem [259] allows us to make a sensitivity analysis. An important result from his first order perturbation theory for simple eigenvalues tells us the following. If \( \lambda \) is a simple
eigenvalue of the pencil $G^{(1)}_t - \lambda G_t$ and $\lambda$ is the corresponding eigenvalue of a
perturbed pencil $\tilde{G}^{(1)}_t - \tilde{\lambda} G_t$ with $\|G^{(1)}_t - \tilde{G}^{(1)}_t\|_2 \approx \|G_t - \tilde{G}_t\|_2 \approx \epsilon$, then
\[
\frac{|\lambda - \lambda_t|}{\sqrt{1 + |\lambda|^2} \sqrt{1 + |\lambda_t|^2}} \leq \frac{\epsilon}{\sqrt{|y^H G^{(1)}_t x|^2 + |y^H G_t x|^2}} + O(\epsilon^2) =: \kappa(\lambda, x, y)\epsilon + O(\epsilon^2)
\]
where $x$ and $y$ are the right and left eigenvectors corresponding to $\lambda$,
\[
G^{(1)}_t x = \lambda G_t \quad \text{and} \quad y^H G^{(1)}_t = \lambda y^H G_t,
\]
normalized such that $\|x\|_2 = \|y\|_2 = 1$. Let us call $\kappa(\lambda, x, y)$ the sensitivity factor
of the eigenvalue $\lambda$.

**Example 15.** Suppose that $n = 10,$
\[
z_1 = -1, \quad z_2 = 4, \quad z_3 = 4 + \delta(1 + i),
\]
\[
z_4 = 3i, \quad z_5 = 3i + \delta(10 + 5i), \quad z_6 = 3i + \delta(-3 + 4i),
\]
\[
z_7 = c + \delta(-1 + 2i), \quad z_8 = c + \delta(1 + 5i), \quad z_9 = c + \delta(1 + i), \quad z_{10} = c + \delta(-2 - 2i),
\]
where $c = -3 + 3i$ and $\delta = 10^{-4}$. Suppose that $\nu_1 = \cdots = \nu_{10} = 0$. Let $f(z)$ be
the polynomial that has $z_1, \ldots, z_{10}$ as simple zeros, $f(z) = \prod_{k=1}^{10} (z - z_k)$, and let
$\gamma = \{ z \in \mathbb{C} : |z| = 5 \}$. Note that $f$ has four clusters of zeros, $m = 4$, of weight 1, 2, 
3 and 4, respectively. We have evaluated the logarithmic derivative of $f(z)$ again via
formula (36). As we do not want the algorithm to stop as soon as it has found the
approximations for the centres of the clusters, we set $\epsilon_{\text{stop}}$ to a rather small value,
$\epsilon_{\text{stop}} = 10^{-18}$. Our algorithm gives the following results. The total number of zeros
is equal to 10. The polynomials $\varphi_0(z)$ and $\varphi_1(z)$ are defined as regular FOPs. The
computed eigenvalues of the pencil $G^{(1)}_t - \lambda G_t$ lead to zeros that lie inside $\gamma$, and
thus the polynomial $\varphi_2(z)$ is defined as a regular FOP. The sensitivity factors of the
eigenvalues are equal to
\[
3.773765047881042e-02 \quad 9.474367914383353e-03
\]
The solution of the Vandermonde system that corresponds to the approximations
for the zeros of $\varphi_0(z)$ is given by
\[
\begin{align*}
7.050510110011339e00 - i \quad & 2.941868186347331e-01 \\
2.949489899988664e00 + i \quad & 2.941868186347336e-01
\end{align*}
\]
The algorithm computes $\langle \varphi_2(z), \varphi_2(z) \rangle$. In step [4] it compares
\[
9.853283227226082e-01
\]
with $\epsilon_{\text{stop}}$ and sets allsmall $\leftarrow$ false. The polynomial $\varphi_2(z)$ is defined as a regular
FOP. The sensitivity factors of the eigenvalues are equal to
\[
5.221220847969128e-03 \quad 5.69118489264763e-02 \quad 9.4403095682826e-03
\]
The solution of the Vandermonde system that corresponds to the approximations
for the zeros of $\varphi_2(z)$ is given by
\[
\begin{align*}
\end{align*}
\]
The algorithm computes $\phi_1(z)$, $\phi_2(z)$. In step [4] it compares

9.560608076054004 e-02

with $\epsilon_{\text{stop}}$ and sets allsmall $\leftarrow$ false. The polynomial $\phi_3(z)$ is defined as a regular FOP. The sensitivity factors of the eigenvalues are equal to

5.97609084456927 e-03
1.635214220936046 e-02
2.194979920293141 e-02
4.92311812602375 e-03

The solution of the Vandermonde system that corresponds to the approximations for the zeros of $\phi_3(z)$ is given by

1.99999998746185 e+00 - i 1.260274035855700 e-08
4.000000263704519 e+00 - i 1.428161173014011 e-07
2.00000000825852 e+00 + i 2.437473983804684 e-07
9.99999997234455 e-01 - i 8.932854039794526 e-08

Observe that these “multiplicities” (actually, they are the weights of the clusters) are at a distance of $O(10^{-8}) = O(\delta^2)$ to integers. This is a first indication of the fact that $m = 4$. The algorithm computes $\langle \phi_4(z), \phi_5(z) \rangle$. In step [4] it compares

4.6092627834357 e-09

with $\epsilon_{\text{stop}}$. As we have given $\epsilon_{\text{stop}}$ a very small value, $\epsilon_{\text{stop}} = 10^{-18}$, the algorithm sets allsmall $\leftarrow$ false and continues. It defines the polynomial $\phi_6(z)$ as a regular FOP. The sensitivity factors of the eigenvalues are equal to

1.859124283121160 e+02
5.97664519337178 e-03
1.632900762729697 e-02
4.930300860498997 e-03
2.20105119773096 e-02

Observe that one of the eigenvalues is much more sensitive than the others. The solution of the Vandermonde system that corresponds to the computed approximations for the zeros of $\phi_6(z)$ is given by

-4.028597029147776 e-08 - i 1.692141150994100 e-07
2.000000072301077 e+00 - i 8.308500282558695 e-09
4.0000000510541128 e+00 - i 2.651932814361151 e-07
1.0000000192451618 e+00 - i 8.61403072692243 e-08
2.99999993062146 e+00 + i 5.288562050783197 e-07

Observe that the component that corresponds to the spurious eigenvalue is of size $O(10^{-8})$ whereas the other components are close to integers. This enables us to deduce the presence of spurious eigenvalues without computing the sensitivity factors. The algorithm computes $\langle \phi_7(z), \phi_8(z) \rangle$. In step [4] it compares

3.544154048709335 e-10

with $\epsilon_{\text{stop}}$ and sets allsmall $\leftarrow$ false. The polynomial $\phi_9(z)$ is defined as a regular FOP. The sensitivity factors of the eigenvalues are equal to

1.751672615711886 e+00 + i 2.391215777821213 e-02
5.761474296212248 e+00 + i 1.41872340124189 e-01
2.4868853086075870 e+00 - i 1.657844817906288 e-01

The algorithm computes $\langle \phi_2(z), \phi_3(z) \rangle$. In step [4] it compares

9.560608076054004 e-02

with $\epsilon_{\text{stop}}$ and sets allsmall $\leftarrow$ false. The polynomial $\phi_4(z)$ is defined as a regular FOP. The sensitivity factors of the eigenvalues are equal to

...
The solution of the Vandermonde system that corresponds to the approximations for the zeros of \( \varphi_0(z) \) is given by

\[
\begin{align*}
-1.109848867030467 \times 10^{-7} - i 3.954458493988473 \times 10^{-8} \\
5.068142899128812 \times 10^{-12} - i 6.003619075497872 \times 10^{-13} \\
2.0000000007044562 \times 10^{0} - i 9.50031341340428 \times 10^{-9} \\
4.000000045529126 \times 10^{0} - i 3.668720544062149 \times 10^{-7} \\
1.000000170753607 \times 10^{0} - i 1.185972639722307 \times 10^{-7} \\
2.999999477890392 \times 10^{0} + i 5.345146309560269 \times 10^{-7}
\end{align*}
\]

The algorithm computes \( \langle \varphi_0(z), \varphi_0(z) \rangle \). In step [4] it compares

\[
1.189012222825083 \times 10^{-9}
\]

with \( \varepsilon_{\text{stop}} \) and sets allsmall \( \leftarrow \) false. And so on. The algorithm defines \( \varphi_7(z) \) as a regular FOP, \( \varphi_8(z) \) as an inner polynomial, \( \varphi_9(z) \) as a regular FOP, and finally \( \varphi_{10}(z) \) as an inner polynomial. The computed approximations for the zeros of \( f \) are given by

\[
\begin{align*}
-9.9999999999027 \times 10^{-1} + i 1.827471507453993 \times 10^{-1} \\
4.0000066008247924 \times 10^{0} + i 5.012986226260452 \times 10^{-5} \\
4.000047628710319 \times 10^{0} - i 2.622572865154105 \times 10^{-4} \\
-5.868580001572310 \times 10^{-0} + i 3.000189455314092 \times 10^{0} \\
1.147726666656671 \times 10^{-3} + i 3.000342939244789 \times 10^{9} \\
-3.000001882960546 \times 10^{0} + i 3.000324337949845 \times 10^{0} \\
-3.000258740452308 \times 10^{0} + i 2.999857462002724 \times 10^{9} \\
-9.085700394437596 \times 10^{-0} + i 1.866967988946470 \times 10^{0} \\
-8.509233181288104 \times 10^{-0} - i 6.684482268880147 \times 10^{0} \\
-4.99930000000002 \times 10^{-0} + i 2.10016000000000 \times 10^{9}
\end{align*}
\]

The relative errors of the approximations for the zeros that belong to the clusters of weight 1 and 2 are \( O(10^{-1}) \) and \( O(10^{-5}) \), respectively. For the other zeros, the relative errors are at least \( O(10^{-3}) \). If we set \( \varepsilon_{\text{stop}} = 10^{-6} \), then our algorithm stops at the polynomial of degree 4.

We obtain the following approximations for the centres of the clusters:

\[
\begin{align*}
-9.99999564181510 \times 10^{-0} - i 5.152524762408461 \times 10^{-8} \\
4.000000001653271 \times 10^{0} + i 5.000694739720757 \times 10^{-5} \\
2.335838430156945 \times 10^{-4} + i 3.000299920075392 \times 10^{0} \\
-3.000024926663507 \times 10^{0} + i 3.000149946356108 \times 10^{0}
\end{align*}
\]

Let us now focus on the separate clusters. We have considered the circles whose centre is the computed approximation for the centre of a cluster and whose radius is equal to 0.1. The relative errors of the approximations for the zeros that we obtain are \( O(10^{-10}), O(10^{-12}), O(10^{-10}) \) and \( O(10^{-6}) \) for the clusters of weight 1, 2, 3 and 4, respectively. If we consider a circle whose centre is the computed approximation for the centre of the cluster of weight 4 and whose radius is equal to \( 10^{-3} \), then
the relative errors of the approximations that we obtain for the zeros that lie in this
cluster are $O(10^{-16})$. Apparently, the smaller the radius of the circle is, the more
accurate the computed approximations for the zeros are. This can be explained by
the fact that the quadrature method gives more accurate approximations for the
integrals (in other words, for the data from which approximations for the zeros are
computed).

More numerical examples will be given in Section 4.

3. Rational interpolation at roots of unity

We will now approach our problem of computing all the zeros of $f$ that lie
inside $\gamma$ in a different way, based on rational interpolation at roots of unity. We will
show how the new approach complements the previous one and how it can be used
effectively in case $\gamma$ is the unit circle. Numerical examples will be given in Section 4.

Let $K$ be a positive integer and let $t_1, \ldots, t_K$ be the $K$th roots of unity,
$$t_k := \exp\left(\frac{2\pi i k}{K}\right), \quad k = 1, \ldots, K.$$ Define $g_{K-1}(z)$ as the polynomial
$$g_{K-1}(z) := s_0 z^{K-1} + s_1 z^{K-2} + \cdots + s_{K-1}.$$ Note that $\deg g_{K-1}(z) = K - 1$ as by assumption $s_0 \neq 0$. Without loss of generality
we may assume that $g_{K-1}(t_k) \neq 0$ for $k = 1, \ldots, K$. (This condition will be needed
in Theorem 23.)

Let $w_K(z) := z^K - 1$ and define the symmetric bilinear form $\langle\langle \cdot, \cdot \rangle\rangle$ as
$$\langle\langle \phi, \psi \rangle\rangle := \sum_{k=1}^{K} \frac{g_{K-1}(t_k)}{w_K(t_k)} \phi(t_k) \psi(t_k)$$ for $\phi, \psi \in \mathcal{P}$. Note that this form can be evaluated via FFT.

Define the ordinary moments $\sigma_p$ associated with the form $\langle\langle \cdot, \cdot \rangle\rangle$ as
$$\sigma_p := \langle\langle 1, z^p \rangle\rangle = \sum_{k=1}^{K} \frac{g_{K-1}(t_k)}{w_K(t_k)} t_k^p$$ for $p = 0, 1, 2, \ldots$ and let $H_k$ be the $k \times k$ Hankel matrix
$$H_k := \begin{bmatrix} \sigma_0 & \sigma_1 & \cdots & \sigma_{k-1} \\ \sigma_1 & \cdots & \cdots & \cdots \\ \vdots & \ddots & \ddots & \ddots \\ \sigma_{k-1} & \cdots & \cdots & \sigma_{2k-2} \end{bmatrix}$$ for $k = 1, 2, \ldots$. Then the regular FOP $f_\tau$ of degree $\tau \geq 1$ associated with the form
$\langle\langle \cdot, \cdot \rangle\rangle$ exists if and only if the matrix $H_\tau$ is nonsingular. Also, the following theorem holds (cf. Theorem 3).

**Theorem 23.** $K = \text{rank } H_{K+p}$ for every nonnegative integer $p$. 
Thus $\mathcal{H}_g$ is nonsingular whereas $\mathcal{H}_f$ is singular for $\tau > K$. The regular FOP $f_K$ of degree $K$ exists while regular FOPs of degree larger than $K$ do not exist. The polynomial $f_K$ is easily seen to be

$$f_K(z) = (z - t_1) \cdots (z - t_K) = w_K(z).$$

It is the monic polynomial of degree $K$ that has $t_1, \ldots, t_K$ as simple zeros.

If $\mathcal{H}_K$ is strongly nonsingular, then we have a full set $\{f_0, f_1, \ldots, f_K\}$ of regular FOPs. Else, we can proceed in the same way as with the form $\langle \cdot, \cdot \rangle$. By filling up the gaps in the sequence of existing regular FOPs it is possible to define a sequence $\{f_\tau\}_{\tau=0}^\infty$, with $f_\tau$ a monic polynomial of degree $\tau$, such that if these polynomials are grouped into blocks according to the sequence of regular indices, then polynomials belonging to different blocks are orthogonal with respect to $\langle \cdot, \cdot \rangle$. More precisely, define $\{f_\tau\}_{\tau=0}^\infty$ as follows. If $\tau$ is a regular index, then let $f_\tau$ be the regular FOP of degree $\tau$. Else define $f_\tau$ as $f_\rho h_\tau$, where $\rho$ is the largest regular index less than $\tau$ and $h_\tau$ is an arbitrary monic polynomial of degree $\tau - \rho$. In the latter case $f_\tau$ is called an inner polynomial. If $f_\rho(z) = z^{\tau - \rho}$ then we say that $f_\tau$ is defined by using the standard monomial basis. The block orthogonality property is expressed by the fact that the Gram matrix $[(f_\tau, f_i)]_{\tau, i=0}^{K-1}$ is block diagonal. The diagonal blocks are nonsingular, symmetric and zero above the main antidiagonal. If all the inner polynomials in a certain block are defined by using the standard monomial basis, then the corresponding diagonal block has Hankel structure, cf. Theorem 5.

The definition of the form $\langle \cdot, \cdot \rangle$ may seem arbitrary. However, there exists a remarkable connection between the forms $\langle \cdot, \cdot \rangle$ and $\langle \cdot, \cdot \rangle$.

**Theorem 24.** Let $\phi, \psi \in \mathcal{P}$. If $\deg \phi + \deg \psi \leq K - 1$, then $\langle \phi, \psi \rangle = \langle \phi, \psi \rangle$.

**Proof.** Let $V(t_1, \ldots, t_K)$ be the Vandermonde matrix with nodes $t_1, \ldots, t_K$,

$$V(t_1, \ldots, t_K) := \begin{bmatrix} 1 & t_1 & \cdots & t_1^{K-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & t_K & \cdots & t_K^{K-1} \end{bmatrix}.$$ 

Then

$$\begin{bmatrix} g_{K-1}(t_1) \\ \vdots \\ g_{K-1}(t_K) \end{bmatrix} = V(t_1, \ldots, t_K) \begin{bmatrix} s_{K-1} \\ \vdots \\ s_0 \end{bmatrix}.$$ 

As $V(t_1, \ldots, t_K)/\sqrt{K}$ is unitary, it follows that

$$\begin{bmatrix} s_{K-1} \\ \vdots \\ s_0 \end{bmatrix} = \frac{1}{K} V(t_1, \ldots, t_K)^H \begin{bmatrix} g_{K-1}(t_1) \\ \vdots \\ g_{K-1}(t_K) \end{bmatrix}.$$ 

As $w_K(z) = z^K - 1$, it follows that $w_K(t_k) = K z^{K-1}$ and thus $w_K(t_k) = K/t_k$ for $k = 1, \ldots, K$. Let $j \in \{1, \ldots, K\}$. Then

$$\sigma_{K-j} = \frac{1}{K} \sum_{k=1}^K g_{K-1}(t_k) t_k^{K-1-j} = \frac{1}{K} \sum_{k=1}^K g_{K-1}(t_k) t_k^{j-1}$$.
and thus

\[
\begin{bmatrix}
\sigma_{K-1} \\
\sigma_{K-2} \\
\vdots \\
\sigma_0
\end{bmatrix} = \frac{1}{K} \begin{bmatrix}
t_1 & \cdots & t_K \\
t_{K-1} & \cdots & t_{K-1} \\
\vdots & & \vdots \\
t_1 & \cdots & t_K
\end{bmatrix} \begin{bmatrix}
g_{K-1}(t_1) \\
g_{K-1}(t_2) \\
\vdots \\
g_{K-1}(t_K)
\end{bmatrix}
\]

\[
= \frac{1}{K} [V(t_1, \ldots, t_K)]^T H
\begin{bmatrix}
g_{K-1}(t_1) \\
\vdots \\
g_{K-1}(t_K)
\end{bmatrix}
\]

\[
= \begin{bmatrix}
s_{K-1} \\
s_{K-2} \\
\vdots \\
s_0
\end{bmatrix}.
\]

In other words, \( s_p = \sigma_p \) for \( p = 0, 1, \ldots, K - 1 \). As \( \langle \phi, \psi \rangle \) depends on \( \sigma_p \) for \( p = 0, 1, \ldots, \deg(\phi) \), this proves the theorem.

**Corollary 25.** Let \( \tau \) be a nonnegative integer. If \( 2\tau - 1 \leq K \), then \( \tau \) is a regular index for \( \langle \cdot, \cdot \rangle \) if and only if \( \tau \) is a regular index for \( \langle \cdot, \cdot \rangle \). Moreover, if \( 2\tau \leq K \) and if \( \tau \) is a regular index, then \( f_\tau(z) \equiv \varphi_\tau(z) \). Else, if \( \tau \) is not a regular index, then \( f_\tau(z) = R_{\tau, \rho}(z) \varphi_\tau(z) \) where \( \rho \) is the largest regular index less than \( \tau \) and \( R_{\tau, \rho}(z) \) is a rational function of type \( \varphi_\tau(z) \). If \( f_\tau(z) \) and \( \varphi_\tau(z) \) are both defined by using the standard monomial basis, then \( R_{\tau, \rho}(z) \equiv 1 \).

**Corollary 26.** If \( K \geq 2n \) and \( n \leq \tau \leq [K/2] \), then \( f_\tau(z_k) = 0 \) for \( k = 1, \ldots, n \) and \( \langle z^p, f_\tau(z) \rangle = 0 \) for all \( p \geq 0 \). Also, \( \langle z^p, f_\tau(z) \rangle = 0 \) for \( p = \tau, \ldots, K - 1 - \tau \). (Note that the latter range may be empty.)

**Corollary 27.** If \( K \geq 2m \) and \( m \leq \tau \leq [K/2] \), then \( f_\tau(c_j) = O(\delta^2) \), \( \delta \to 0 \) for \( j = 1, \ldots, m \) and \( \langle z^p, f_\tau(z) \rangle = O(\delta^2), \delta \to 0 \) for all \( p \geq \tau \). Also, \( \langle z^p, f_\tau(z) \rangle = O(\delta^2), \delta \to 0 \) for \( p = \tau, \ldots, K - 1 - \tau \). (Note that the latter range may be empty.)

Thus, if \( K \geq 2N \), then we can apply the algorithm that we have presented in Chapter 1 to the form \( \langle \cdot, \cdot \rangle \) and we will obtain exactly the same results as with the form \( \langle \cdot, \cdot \rangle \). This is an interesting fact in its own right. The main reason, though, that motivated us to introduce the form \( \langle \cdot, \cdot \rangle \) is the fact that it is related to rational interpolation. We will show that the “denominator polynomials” in a certain linearized rational interpolation problem that is related to the polynomial \( g_{K-1}(z) \) are FOPs with respect to \( \langle \cdot, \cdot \rangle \). This will lead to an alternative way to calculate the FOPs \( f_\tau(z) \) and thus, because of Corollary 25, the FOPs \( \varphi_\tau(z) \).

Let \( \sigma \) and \( \tau \) be nonnegative integers such that \( \sigma + \tau + 1 = K \). Let \( p_\tau(z) \) and \( q_\tau(z) \) be polynomials, where

\[
\deg p_\tau(z) \leq \sigma \quad \text{and} \quad \deg q_\tau(z) \leq \tau,
\]

such that the following linearized rational interpolation conditions are satisfied:

\[
\begin{align*}
p_\tau(t_k) - q_\tau(t_k) g_{K-1}(t_k) &= 0, \\
k &= 1, \ldots, K.
\end{align*}
\]
Each pair of polynomials \((p_\sigma(z), q_\tau(z))\) that satisfies the degree conditions (40) and the interpolation conditions (41) is called a multipoint Padé form (MPF). The polynomials \(p_\sigma(z)\) and \(q_\tau(z)\) will be called numerator polynomial and denominator polynomial, respectively.

The interpolation conditions (41) lead to a system of \(K\) linear equations in \(K + 1\) unknowns, and thus at least one nontrivial (i.e., whose numerator and denominator polynomial are not identically equal to zero) MPF exists. As (41) are homogeneous linear equations, every scalar multiple of a MPF is also a MPF. From now on, we will always assume that MPFs are normalized such that the denominator polynomial is monic. However, the fact that then the number of interpolation conditions is equal to the number of unknown polynomial coefficients, does not guarantee that there exists only one MPF. It merely guarantees that every MPF leads to the same irreducible rational function, called multipoint Padé approximant (MPA). Indeed, suppose that there exist two MPAs. Then the numerator polynomial of the difference of these MPAs is a polynomial of degree \(\leq \sigma + \tau\) that vanishes at \(\sigma + \tau + 1\) points. This numerator polynomial is therefore identically equal to zero, which implies that the MPA is unique.

Let \(\mathcal{R}_{\sigma,\tau}\) be the set of rational functions of type \([\sigma/\tau]\), i.e., with numerator degree at most \(\sigma\) and denominator degree at most \(\tau\). A rational interpolation problem that is closely related to (41) is the Cauchy interpolation problem: find all irreducible rational functions \(r_{\sigma,\tau}(z) \in \mathcal{R}_{\sigma,\tau}\) whose denominator polynomial is monic, such that

\[
r_{\sigma,\tau}(t_k) = g_{K-1}(t_k), \quad k = 1, \ldots, K.
\]

This interpolation problem is not always solvable. If a solution exists, then it is unique, and it is equal to the MPA. In general, however, the MPA need not solve the Cauchy interpolation problem: the numerator and denominator polynomials of the MPFs may have common zeros at some interpolation points. The MPA may not satisfy the interpolation condition (42) at these points, which are then called unattainable points.

Let \(r(z) \in \mathcal{R}_{\sigma,\tau}\) and suppose that \(r(z) = p(z)/q(z)\) where \(p(z)\) and \(q(z)\) are relatively prime polynomials. The \textit{defect} of \(r\) with respect to \(\mathcal{R}_{\sigma,\tau}\) is then defined as

\[
\min\{ \sigma - \deg p(z), \tau - \deg q(z) \}.
\]

The following theorem provides the general solution of the linearized rational interpolation problem.

**Theorem 28.** The general MPA that corresponds to the degree conditions (40) and the interpolation conditions (41) is given by

\[
(p_\sigma(z), q_\tau(z)) = \left( \tilde{p}_\sigma(z)s(z)u(z), \tilde{q}_\tau(z)s(z)u(z) \right),
\]

where \(\tilde{p}_\sigma(z)\), \(\tilde{q}_\tau(z)\) and \(s(z)\) are uniquely determined polynomials, and where \(u(z)\) is arbitrary. The polynomials \(\tilde{p}_\sigma(z)\) and \(\tilde{q}_\tau(z)\) are relatively prime, and \(s(z)\) is a divisor of \(w_K(z)\). Let \(\delta_{\sigma,\tau}\) be the defect of \(\tilde{p}_\sigma(z)/\tilde{q}_\tau(z)\) with respect to \(\mathcal{R}_{\sigma,\tau}\). Then

\[
\deg s(z) \leq \delta_{\sigma,\tau}\text{ and } \deg u(z) \leq \delta_{\sigma,\tau} - \deg s(z).
\]

The zeros of \(s(z)\) are the unattainable points for the corresponding Cauchy interpolation problem.

**Proof.** See, for example, Gutknecht [124, p. 549].
The literature on rational interpolation is vast. We will not give a full account of all the other issues (in particular, the block structure of the Newton-Padé table) that are involved. The reader may wish to consult the papers by Meinguet [212], Antoulas [13, 14, 15], Berrut and Mittelmann [31] or Gutknecht [124, 125, 126, 127], and the references cited therein.

What is of special interest to us, is the fact that the denominator polynomials \( q_r(z) \) are formal orthogonal polynomials with respect to \( \langle \cdot, \cdot \rangle \).

**Theorem 29.** Let \( \sigma \) and \( \tau \) be nonnegative integers such that \( \sigma + \tau + 1 = K \). Let \( (p_r(z), q_r(z)) \) be a MPF for the degree conditions (40) and the interpolation conditions (41). Then \( \langle (z^p, q_r(z)) \rangle = 0 \) for \( p = 0, 1, \ldots, K - 2 - \deg p_r(z) \) and \( \langle (z^p, q_r(z)) \rangle \neq 0 \) if \( p = K - 1 - \deg p_r(z) \).

**Proof.** Apparently this orthogonality relation was already known to Jacobi. As it plays a very important role in our approach, we prefer to give a (short, but explicit) proof. See also Eğecioğlu and Koç [73] and Gemignani [100] for a slightly weaker version of this theorem.

Let \( p \in \{0, 1, \ldots, K - 2 - \deg p_r(z)\} \). Then

\[
(43) \sum_{k=1}^{K} \frac{t_k^p p_r(t_k)}{w_K(t_k)} = \sum_{k=1}^{K} \frac{g_{K-1}(t_k)}{w'_K(t_k)} t_k^p q_r(t_k)
\]

and \( z^p p_r(z) \) is a polynomial of degree \( p + \deg p_r(z) \leq K - 2 \). Lagrange's formula for the polynomial \( y_{K-1}(z) \) of degree \( \leq K - 1 \) that interpolates the polynomial \( z^p p_r(z) \) in the points \( t_1, \ldots, t_K \) implies that the left hand side of (43) is equal to the coefficient of \( z^{K-1} \) of \( y_{K-1}(z) \). As \( \deg z^p p_r(z) < K - 1 \), it follows that \( y_{K-1}(z) = z^p p_r(z) \) and that the coefficient of \( z^{K-1} \) of \( y_{K-1}(z) \) is equal to zero. It follows that \( \langle (z^p, q_r(z)) \rangle = 0 \) for \( p = 0, 1, \ldots, K - 2 - \deg p_r(z) \). A similar reasoning shows that \( \langle (z^p, q_r(z)) \rangle \neq 0 \) if \( p = K - 1 - \deg p_r(z) \). This proves the theorem. \( \square \)

The following theorem implies that the coefficients (in the standard monomial basis) of the numerator polynomial \( p_r(z) \) of a MPF \( (p_r(z), q_r(z)) \) that correspond to the degree conditions (40) and the interpolation conditions (41) can be expressed as inner products with respect to \( \langle \cdot, \cdot \rangle \). This explains how the degree property \( \deg p_r(z) \leq \sigma \) is related to the formal orthogonality property satisfied by \( q_r(z) \).

**Theorem 30.** Suppose that \( q(z) \) is a polynomial and let \( p(z) \) be the polynomial of degree \( \leq K - 1 \) that interpolates \( g_{K-1}(z)q(z) \) at the points \( t_1, \ldots, t_K \). Let \( p(z) = p_0 + p_1 z + \cdots + p_{K-1} z^{K-1} \). Then \( p_k = \langle (z^{K-1-k}, q(z)) \rangle \) for \( k = 0, 1, \ldots, K - 1 \).

**Proof.** The Lagrange representation of \( p(z) \) is given by

\[
p(z) = \sum_{k=1}^{K} \pi_k L_k(z)
\]

where

\[
\pi_k := g_{K-1}(t_k)q(t_k) w'_k(z) \quad \text{and} \quad L_k(z) := \frac{w_K(z)}{z - t_k}
\]
for \( k = 1, \ldots, K \). Let \( L_k(z) = L_{0,k} + L_{1,k}z + \cdots + L_{K-1,k}z^{K-1} \) for \( k = 1, \ldots, K \).

Note that \( L_{K-1,1} = \cdots = L_{K-1,K} = 1 \). Let

\[
V := \begin{bmatrix}
1 & t_1 & \cdots & t_1^{K-1} \\
\vdots & \cdots & \cdots & \vdots \\
1 & t_K & \cdots & t_K^{K-1}
\end{bmatrix}
\]

be the Vandermonde matrix with nodes \( t_1, \ldots, t_K \) and let

\[
L := \begin{bmatrix}
L_{0,1} & \cdots & L_{0,K} \\
\vdots & \cdots & \vdots \\
L_{K-1,1} & \cdots & L_{K-1,K}
\end{bmatrix}
\]

be the matrix that contains the coefficients of \( L_1(z), \ldots, L_K(z) \). Then

\[
VL = \text{diag}(L_1(t_1), \ldots, L_K(t_K))
\]

\[
= \text{diag}(w'_K(t_1), \ldots, w'_K(t_K))
\]

\[
= K \text{ diag}(\overline{t_1}, \ldots, \overline{t_K}).
\]

As \( V/\sqrt{K} \) is unitary, it follows that \( V^{-1} = V^H/K \) and thus

\[
L = V^H \text{ diag}(\overline{t_1}, \ldots, \overline{t_K}) = \begin{bmatrix}
t_1^{K-1} & \cdots & t_K^{K-1} \\
\vdots & \cdots & \vdots \\
t_1 & \cdots & t_K \\
1 & \cdots & 1
\end{bmatrix}.
\]

In other words, \( L_{j,k} = t_k^{K-1-j} \) for \( k = 1, \ldots, K \) and \( j = 0, 1, \ldots, K - 1 \). As

\[
p_j = \sum_{k=1}^K L_{j,k} \pi_k
\]

for \( j = 0, 1, \ldots, K - 1 \), it follows that

\[
p_j = \sum_{k=1}^K \frac{q_{K-1}(t_k)}{w'_K(t_k)} t_k^{K-1-j} q(t_k) = \langle \langle z^{K-1-j}, q(z) \rangle \rangle
\]

for \( j = 0, 1, \ldots, K - 1 \). This proves the theorem. \( \square \)

The following theorem shows how to construct the sequence of FOPs \( f_\tau(z) \) from the MPFs \( (p_\tau(z), q_\tau(z)) \). Regular FOPs correspond to denominator polynomials whose degree is equal to \( \tau \) whereas the sizes of the blocks are determined by the actual degrees of the numerator polynomials.

**Theorem 31.** Let \( \sigma \) and \( \tau \) be nonnegative integers such that \( \sigma + \tau + 1 = K \). Let \( (p_\tau(z), q_\tau(z)) = (\hat{p}_\tau(z)s(z)u(z), \hat{q}_\tau(z)s(z)u(z)) \) be the general MPF for the degree conditions (40) and the interpolation conditions (41), where the polynomials \( \hat{p}_\tau(z) \), \( \hat{q}_\tau(z) \), \( s(z) \) and \( u(z) \) are as in Theorem 28. If \( \tau \) is a regular index for \( \langle \langle \cdot, \cdot \rangle \rangle \), then \( \deg(\hat{q}_\tau(z)s(z)) = \tau \), the FOP \( f_\tau(z) \) is given by \( f_\tau(z) \equiv \hat{q}_\tau(z)s(z) \) and the smallest regular index that is larger than \( \tau \) is equal to \( K - \deg(\hat{p}_\tau(z)s(z)) \). Conversely, if \( \deg(\hat{q}_\tau(z)s(z)) = \tau \), then \( \tau \) is a regular index for \( \langle \langle \cdot, \cdot \rangle \rangle \).

**Proof.** Suppose that \( \tau \) is a regular index for \( \langle \langle \cdot, \cdot \rangle \rangle \). Then \( \det \mathcal{H}_\tau \neq 0 \) and there exists precisely one monic polynomial \( f_\tau(z) \) of degree \( \tau \) such that

\[
\langle \langle z^p, f_\tau(z) \rangle \rangle = 0 \quad \text{for} \quad p = 0, 1, \ldots, \tau - 1.
\]
Let \( p(z) \) be the polynomial of degree \( \leq K - 1 \) that interpolates \( g_{K-1}(z)f_r(z) \) at the points \( t_1, \ldots, t_K \). Then, according to Theorem 30, \( \deg p(z) \leq K - \tau - 1 = \sigma \). Thus \((p(z), f_r(z))\) is a MPF for the degree conditions (40) and the interpolation conditions (41). In other words, there exists a MPF whose denominator polynomial has degree \( \tau \). Theorem 28 then implies that there exists a monic polynomial \( u_r(z) \) of degree \( \tau - \deg(\hat{q}_r(z)s(z)) \) such that \( f_r(z) \equiv \hat{q}_r(z)s(z)u_r(z) \). If \( \deg u_r(z) > 0 \), then we can choose a different monic polynomial \( \tilde{u}_r(z) \) of the same degree. Then \( f_r(z) \not= \hat{q}_r(z)s(z)\tilde{u}_r(z) \) and, by Theorem 29,

\[
\langle (z^p, \hat{q}_r(z)s(z)\tilde{u}_r(z)) \rangle = 0 \quad \text{for} \quad p = 0, 1, \ldots, \tau - 1.
\]

As \( \deg(\hat{q}_r(z)s(z)\tilde{u}_r(z)) = \tau \), this contradicts the fact that \( f_r(z) \) is unique. Thus we may conclude that \( \deg(\hat{q}_r(z)s(z)) = \tau \) and \( f_r(z) \equiv \hat{q}_r(z)s(z) \). Now Theorem 29 implies that

\[
\langle (z^p, f_r(z)) \rangle = 0 \quad \text{for} \quad p = 0, 1, \ldots, K - 2 - \deg(\hat{p}_r(z)s(z))
\]

and

\[
\langle (z^p, f_r(z)) \rangle \neq 0 \quad \text{if} \quad p = K - 1 - \deg(\hat{p}_r(z)s(z))\]

The structure of the diagonal blocks of the Gram matrix \([\langle f_r, f_t \rangle]_{t=0}^{K-1}\) then implies that \( \det H_t = 0 \) for \( t = \tau + 1, \ldots, K - 1 - \deg(\hat{p}_r(z)s(z)) \) and \( \det H_t \neq 0 \) if \( t = K - \deg(\hat{p}_r(z)s(z)) \).

Suppose that \( \deg(\hat{q}_r(z)s(z)) = \tau \). Then there exists only one MPF for the degree conditions (40) and the interpolation conditions (41). The polynomial \( \hat{q}_r(z)s(z) \) is a monic polynomial of degree \( \tau \) and, according to Theorem 29,

\[
\langle (z^p, \hat{q}_r(z)s(z)) \rangle = 0 \quad \text{for} \quad p = 0, 1, \ldots, \tau - 1.
\]

Suppose that there exists another monic polynomial \( \tilde{f}_r(z) \) of degree \( \tau \), \( \tilde{f}_r(z) \not= \hat{q}_r(z)s(z) \), such that

\[
\langle (z^p, \tilde{f}_r(z)) \rangle = 0 \quad \text{for} \quad p = 0, 1, \ldots, \tau - 1.
\]

Let \( p(z) \) be the polynomial of degree \( \leq K - 1 \) that interpolates \( g_{K-1}(z)f_r(z) \) at the points \( t_1, \ldots, t_K \). Then, according to Theorem 30, \( \deg p(z) \leq K - \tau - 1 = \sigma \). Thus \((p(z), \tilde{f}_r(z))\) is a MPF for the degree conditions (40) and the interpolation conditions (41). It follows that \( \tilde{f}_r(z) \equiv \hat{q}_r(z)s(z) \). In other words, there exists only one monic polynomial \( \tilde{f}_r(z) \) of degree \( \tau \) that is orthogonal (with respect to \( \langle \cdot, \cdot \rangle \)) to all polynomials of lower degree. Thus \( \tau \) is a regular index for \( \langle \cdot, \cdot \rangle \). This proves the theorem.

The previous theorem suggests the following look-ahead strategy. Start with \( \tau = 0 \) and the corresponding MPF \((g_{K-1}(z), 1)\). Then set \( \tau \leftarrow K - \deg g_{K-1}(z) \). Note that \( \tau = 1 \) as \( \deg g_{K-1}(z) = K - 1 \). Compute the corresponding MPF \((p_r(z), q_r(z))\). Note that, as \( \tau \) is a regular index for \( \langle \cdot, \cdot \rangle \), this MPF is uniquely defined, i.e., the polynomial \( u(z) \equiv 1 \) (cf. Theorem 28). Use \( K - \deg p_r(z) \) as the next value of \( \tau \), and so on. Observe that, if \( \deg p_r(z) = \sigma \), then the next value of \( \tau \) is given by \( \tau + 1 \). The interpolation problems can be solved via the algorithm of Van Barel and Bultheel [274]. This algorithm provides the coefficients of the numerator and the denominator polynomial in the standard monomial basis.
Of course, in floating-point arithmetic this strategy will only work if one uses a concept of ‘numerical degree’ instead of the classical ‘degree’. The numerical degree of a polynomial can be defined as follows. Let \( \epsilon > 0 \). The \( \epsilon \)-degree of a polynomial
\[
p(z) = p_0 + p_1 z + \cdots + p_{K-1} z^{K-1} \in \mathcal{P}
\]
of degree \( \leq K - 1 \) is defined as follows. Let
\[
\chi(k) := \max \{ |p_{k+1}|, \ldots, |p_{K-1}| \} / |p_k|
\]
for all \( k \in \{0, 1, \ldots, K - 2\} \) such that \( p_k \neq 0 \) and \( \chi(k) = \infty \) otherwise. If
\[
\min_{0 \leq k \leq K-2} \chi(k) \leq \epsilon,
\]
then the \( \epsilon \)-degree of \( p(z) \) is defined as the index \( k \) for which the minimum in (44) is attained. Else, the \( \epsilon \)-degree of \( p(z) \) is set equal to \( K - 1 \).

The following corollaries provide us with a stopping criterion.

**Corollary 32.** Let \( \sigma \) and \( \tau \) be nonnegative integers such that \( \sigma + \tau + 1 = K \). Let \( (p_r(z), q_r(z)) \) be a MPF for the degree conditions (40) and the interpolation conditions (41). If \( K \geq 2n \) and \( n \leq \tau \leq [K/2] \), then \( \deg p_r(z) \leq \deg q_r(z) - 1 \).

**Proof.** This follows from Theorem 30, Theorem 24, and Corollary 26.

**Corollary 33.** Let \( \sigma \) and \( \tau \) be nonnegative integers such that \( \sigma + \tau + 1 = K \). Let \( (p_r(z), q_r(z)) \) be a MPF for the degree conditions (40) and the interpolation conditions (41). Let \( p_r(z) = p_0 + p_1 z + \cdots + p_{K-1} z^{K-1} \). If \( K \geq 2m \) and \( m \leq \tau \leq [K/2] \), then \( p_k = \mathcal{O}(\delta^2) \), \( \delta \to 0 \) for \( k = \deg q_r(z), \ldots, K - 1 \). In other words, if \( \epsilon \) is sufficiently small, then the \( \epsilon \)-degree of \( p_r(z) \) is less or equal than \( \deg q_r(z) - 1 \).

**Proof.** This follows from Theorem 30, Theorem 24, and Corollary 27.

In other words, at the end the (numerical) degree of the numerator polynomial is less or equal than the degree of the denominator polynomial minus one. One can easily verify that this stopping criterion is equivalent to the one used in the algorithm that we have presented in Chapter 1.

Let us consider the problem of how to evaluate the polynomial \( g_{K-1}(z) \) at the \( K \)th roots of unity \( t_1, \ldots, t_K \). One can easily verify that
\[
g_{K-1}(z) = \frac{1}{2\pi i} \int_{\gamma} \frac{t^K - z^K}{t - z} f(t) \frac{d t}{f(t)}
\]
if \( z \notin \gamma \). Thus, if \( t_k \notin \gamma \) for \( k = 1, \ldots, K \), then
\[
g_{K-1}(t_k) = \frac{1}{2\pi i} \int_{\gamma} \frac{t^K - 1}{t - t_k} f(t) \frac{d t}{f(t)}
\]
for \( k = 1, \ldots, K \).

In case \( \gamma \) is the unit circle, one can obtain accurate approximations for
\[
g_{K-1}(t_1), \ldots, g_{K-1}(t_K)
\]
in a very efficient way. Let \( L \) be a positive integer \( \geq K \), preferably a power of 2. Let \( \omega_1, \ldots, \omega_L \) be the \( L \)th roots of unity,
\[
\omega_l := \exp \left( \frac{2\pi i}{L} \right), \quad l = 1, \ldots, L.
\]
Theorem 34. Suppose that $\gamma$ is the unit circle. Let $v_L \in \mathbb{C}^{L \times 1}$ be the vector
\[ v_L := \frac{1}{L} \begin{bmatrix} 1 & \omega_1 & \cdots & \omega^{L-1}_1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_L & \cdots & \omega^{L-1}_L \end{bmatrix}^H \begin{bmatrix} (f'/f)(\omega_1) \\ \vdots \\ (f'/f)(\omega_L) \end{bmatrix}. \]

Then
\[ [O_{K \times (L-K)} I_K] v_L \approx \begin{bmatrix} s_{K-1} \\ \vdots \\ s_0 \end{bmatrix} \]

where $O_{K \times (L-K)}$ denotes the $K \times (L - K)$ zero matrix and $I_K$ denotes the $K \times K$ identity matrix. In other words, the $K$ last components of $v_L$ are approximations for $s_{K-1}, \ldots, s_1, s_0$.

Proof. By approximating
\[ s_p = \frac{1}{2\pi i} \int_\gamma z^p \frac{f'(z)}{f(z)} \, dz = \frac{1}{2\pi} \int_0^{2\pi} e^{ip\theta} e^{i\theta} \frac{f'(e^{i\theta})}{f(e^{i\theta})} \, d\theta, \quad p = 0, 1, 2, \ldots, \]
via the trapezoidal rule, we obtain that
\[ s_p \approx \frac{1}{L} \sum_{l=1}^L \omega_l^p \frac{f'(\omega_l)}{f(\omega_l)}, \quad p = 0, 1, 2, \ldots. \]

It follows that
\[ s_p \approx \frac{1}{L} \sum_{l=1}^L \omega_l^{p-1} \frac{f'(\omega_l)}{f(\omega_l)}, \quad p = 0, 1, \ldots, L - 1. \]

This proves the theorem. $\square$

Since
\[ \begin{bmatrix} g_{K-1}(t_1) \\ \vdots \\ g_{K-1}(t_K) \end{bmatrix} = \begin{bmatrix} 1 & t_1 & \cdots & t_1^{K-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & t_K & \cdots & t_K^{K-1} \end{bmatrix} \begin{bmatrix} s_{K-1} \\ \vdots \\ s_0 \end{bmatrix}, \]

it follows that we can obtain approximations for $g_{K-1}(t_1), \ldots, g_{K-1}(t_K)$ via one $L$-point (inverse) FFT and one $K$-point FFT.

4. More numerical examples

We have implemented the strategy described in the previous section in Matlab. We have considered the case that $\gamma$ is the unit circle. Approximations for $g_{K-1}(t_1), \ldots, g_{K-1}(t_K)$ have been computed by using Theorem 34. The interpolation problems have been solved via the algorithm of Van Barel and Bultheel [274].
Example 16. Let us reconsider the problem that we have studied in Example 15. As the corresponding $\gamma$ is given by $\gamma = \{ z \in \mathbb{C} : |z| = 5 \}$, we divide all the zeros by 5 to transform the problem to the unit disk. Recall that $N = n = 10$ whereas $m = 4$. We set $L = 512$ and $K = 22$.

In Figure 1 we plot the logarithm with base 10 of the modulus of the coefficients of $p_\tau(z)$ for $\tau = 0, 1, \ldots, 5$. (The logarithm of the modulus of the lowest degree coefficient is shown on the left. In general, the coefficient of $z^k$ corresponds to the abscissa $k + 1$.) Note that $\sigma = 21 - \tau$.

![Figure 1](image_url)

**Figure 1.** The coefficients of $p_\tau(z)$ for $\tau = 0, 1, \ldots, 5$
Clearly \( m = 4 \). By multiplying the zeros of \( q_4(z) \), as computed via the Matlab command \texttt{roots}, by 5 to transform them back to the setting of Example 15, we obtain the following:

-9.99999564178451e-01 - i 5.152536484956870e-08
4.00005001653282e+00 + i 5.000694740214643e-05
2.335838430343322e-04 + i 3.000299920075351e+00
-3.000024926663527e+00 + i 3.000149946356137e+00

These values are to be compared with the approximations for the centres of the clusters that we have obtained in Example 15, namely

-9.99999564181510e-01 - i 5.152524762408461e-08
4.00005001653271e+00 + i 5.000694739720757e-05
2.335838430156945e-04 + i 3.000299920075392e+00
-3.000024926663507e+00 + i 3.000149946356108e+00

The figure that corresponds to \( \tau = 5 \) is included to illustrate that the results given in Corollary 33 hold not only for \( \tau = m \) but for \( \tau \geq m \). The zeros of \( q_\tau(z) \) lead to the same approximations for the centres as the zeros of \( q_4(z) \) and one spurious “centre.”

**Example 17.** Let

\[
f(z) = (\sinh(2z^2) + \sinh(10z) - 1) \times (\sinh(2z^2) + \sinh(10z) - 1.01) |\sinh(2z^2) + \sinh(10z) - 1.02|.
\]

This function has 21 simple zeros inside the unit circle. They form 7 clusters, where each cluster consists of 3 zeros. Thus \( N = n = 21 \) and \( m = 7 \). This example was also studied by Sakurai et al. [248]. We set \( L = 512 \) and \( K = 42 \).

In Figure 2 we plot the logarithm with base 10 of the modulus of the coefficients of \( p_\tau(z) \) for \( \tau = 0, 1, \ldots, 7 \).

The zeros of \( q_7(z) \) are given by

-1.848537713183581e-01 - i 8.949141853554533e-01
-1.848537713183412e-01 + i 8.949141853554334e-01
-1.003354151041395e-01 - i 3.061151582728444e-01
-1.003354151030711e-01 + i 3.061151582802838e-01
1.335489810139705e-01 - i 6.084120926164355e-01
1.33548981013479e-01 + i 6.084120926165633e-01
8.777826151937687e-02 + i 8.943042856595357e-12

These match the approximations for the centres of the clusters that were given in [248].

\[\Diamond\]

**Acknowledgements**

The results in this chapter were obtained in collaboration with Marc Van Barel.

Tetsuya Sakurai contributed to Theorem 19, 22 and 24. I met Tetsuya in person while he was staying at the Department of Mathematics and Computer Science of the University of Antwerp. I would like to thank Annie Cuyt for inviting us both to give a talk at her colloquium “Computer Arithmetic and Numerical Techniques” (October 23, 1997).
Figure 2. The coefficients of $p_{\tau}(z)$ for $\tau = 0,1,\ldots,7$
Zeros and poles of meromorphic functions

In the previous chapters we have considered the problem of computing all the zeros of an analytic function that lie in the interior of a Jordan curve. We have seen how the algorithm that we have proposed can also be used to locate clusters of zeros. Our algorithm gives accurate results. It is based on the theory of formal orthogonal polynomials. Information concerning the location of the zeros is obtained via numerical integration.

We will show how these results can be used to tackle the problem of computing all the zeros and poles of a meromorphic function that lie in the interior of a Jordan curve. More precisely, given a meromorphic function \( f \) and a positively oriented Jordan curve that does not pass through any zero or pole of \( f \), we will present an algorithm for computing all the zeros and poles of \( f \) that lie inside \( \gamma \), together with their respective multiplicities and orders. An upper bound for the total number of poles of \( f \) that lie inside \( \gamma \) is assumed to be known. Initial approximations for the zeros and poles are not needed.

This chapter is inspired by our paper [191].

1. Introduction

Let \( W \) be a simply connected region in \( \mathbb{C} \), \( f : W \to \mathbb{C} \) analytic in \( W \), and \( \gamma \) a positively oriented Jordan curve in \( W \) that does not pass through any zero of \( f \). In Chapter 1 we have considered the problem of computing all the zeros \( z_1, \ldots, z_n \) of \( f \) that lie inside \( \gamma \), together with their respective multiplicities \( \nu_1, \ldots, \nu_n \). The number of mutually distinct zeros of \( f \) that lie inside \( \gamma \) is denoted by \( n \) while \( N \) stands for the total number of zeros of \( f \) that lie inside \( \gamma \), \( N = \nu_1 + \cdots + \nu_n \). The algorithm that we have presented is based on the theory of formal orthogonal polynomials associated with the symmetric bilinear form

\[
\langle \cdot, \cdot \rangle : \mathcal{P} \times \mathcal{P} \to \mathbb{C} : (\phi, \psi) \mapsto \langle \phi, \psi \rangle := \frac{1}{2\pi i} \int_{\gamma} \phi(z)\overline{\psi(z)} \frac{f'(z)}{f(z)} dz.
\]

As the logarithmic derivative \( f'/f \) has a simple pole at \( z_k \) with residue \( \nu_k \) for \( k = 1, \ldots, n \), Cauchy's Theorem implies that

\[
\langle \phi, \psi \rangle = \sum_{k=1}^{n} \nu_k \phi(z_k)\overline{\psi(z_k)}.
\]  
(45)

Our algorithm proceeds as follows. It computes \( z_1, \ldots, z_n \) as the zeros of the \( n \)th degree formal orthogonal polynomial (FOP) that is associated with \( \langle \cdot, \cdot \rangle \). The value of \( n \) is determined indirectly. Once \( n \) and \( z_1, \ldots, z_n \) have been found, the problem
becomes linear and \( \nu_1, \ldots, \nu_n \) are calculated by solving a Vandermonde system. The total number of zeros \( N \) can be computed via numerical integration. It is an upper bound for \( n \). The fact that such an upper bound is available plays a crucial role in the determination of the value of \( n \).

Now suppose that \( f \) is not analytic but meromorphic in \( W \) and suppose that \( f \) has neither zeros nor poles on \( \gamma \). We will show how the algorithm that we have presented in Chapter 1 can be extended to compute all the zeros and poles of \( f \) that lie in the interior of \( \gamma \), together with their respective multiplicities and orders. Let \( P \) denote the total number of poles of \( f \) that lie inside \( \gamma \) (i.e., the number of poles where each pole is counted according to its order). Suppose from now on that \( N + P > 0 \). Let \( p \) denote the number of mutually distinct poles of \( f \) that lie inside \( \gamma \). Let \( y_1, \ldots, y_p \) be these poles and \( \mu_1, \ldots, \mu_p \) their respective orders. An easy calculation shows that \( f'/f \) has a simple pole at \( y_l \) with residue \( -\mu_l \) for \( l = 1, \ldots, p \). Therefore

\[
\langle \phi, \psi \rangle = \sum_{k=1}^{n} \nu_k \phi(z_k)\psi(z_k) - \sum_{l=1}^{p} \mu_l \phi(y_l)\psi(y_l).
\]

This expression is of the same type as (45). The theorems that we have proven in Chapter 1 do not rely on the fact that the \( \nu_k \)'s are known to be positive integers. They merely use the fact that \( \nu_k \neq 0 \) for \( k = 1, \ldots, n \). It is therefore likely that these results can be extended to cover (46). We will see that \( z_1, \ldots, z_n \) and \( y_1, \ldots, y_p \) can indeed be calculated as the zeros of the FOP of degree \( n + p \). Provided, of course, that an upper bound \( M \) for \( m := n + p \) is known. Assume that an upper bound \( \bar{P} \) for \( P \) is known and define the ordinary moments

\[
s_r := \langle 1, z^r \rangle
\]

for \( r = 0, 1, 2, \ldots \). Then

\[
s_r = \nu_1 z_1^r + \cdots + \nu_n z_n^r - \mu_1 y_1^r - \cdots - \mu_p y_p^r
\]

for \( r = 0, 1, 2, \ldots \). In particular, \( s_0 = N - P \) and we may assume that the value of \( s_0 \) has been computed. As \( n + p \leq N + P = s_0 + 2\bar{P} \leq s_0 + 2\bar{P} \), it follows that we may take \( M = s_0 + 2\bar{P} \). In case \( \gamma \) is the unit circle, an upper bound for \( P \) can be obtained by using the heuristic approach of Gleyse and Kaliaguine [103].

For previous attempts to tackle the problem of computing all the zeros and poles of a meromorphic function that lie in the interior of a Jordan curve, we refer to Abd-Elall, Delvès and Reid [1] and also Ioakimidis [156, 157].

2. Theoretical considerations and numerical algorithm

Instead of (46) we will consider the following even more general setting. Let \( \langle \cdot, \cdot \rangle : \mathcal{P} \times \mathcal{P} \to \mathbb{C} \) be any symmetric and bilinear form such that

\[
\langle \phi, \psi \rangle = \sum_{k=1}^{m} \lambda_k \phi(x_k)\psi(x_k)
\]

for any \( \phi, \psi \in \mathcal{P} \), where \( m \in \mathbb{N}_0 \), \( \lambda_1, \ldots, \lambda_m \in \mathbb{C}_0 \) and \( x_1, \ldots, x_m \in \mathbb{C} \) are unknown. Suppose that the points \( x_1, \ldots, x_m \) are known to be mutually distinct. Assume that
we have an ‘oracle’ at our disposal that provides us with the value of \( \langle \phi, \psi \rangle \) for any \( \phi, \psi \in \mathcal{P} \) upon simple request. Let an upper bound \( M \) for \( m \) be given.

Our unknowns \( m, x_1, \ldots, x_m \) and \( \lambda_1, \ldots, \lambda_m \) can be calculated in the same way as \( n, z_1, \ldots, z_n \) and \( \nu_1, \ldots, \nu_n \) have been calculated in Chapter 1. The key point is that the FOP of degree \( m \) that is associated with the form \( \langle \cdot, \cdot \rangle \) is given by

\[
(z - x_1) \cdots (z - x_m),
\]

as one can easily verify.

Let us start by formulating what could be called the “theoretical” solution, cf. Theorems 3 and 4 and Corollary 8. This will ultimately lead to a numerical algorithm that generalizes the algorithm that we have presented in Chapter 1.

Theorems 3 and 4 have been formulated in terms of ordinary moments and Hankel matrices whereas Corollary 8 relies upon formal orthogonal polynomials. These results can be formulated in terms of polynomials from an arbitrary basis whereas Corollary 8 relies upon simple request. Let an upper bound

\[
M := \max_{r,s} \left\{ \left| \sum_{k=1}^{m} \lambda_k \right| \right\},
\]

for \( m = 1, 2, \ldots \).

The following theorem characterizes \( m \).

**Theorem 35.** \( m = \text{rank} F_{m+r} \) for every nonnegative integer \( r \). In particular, \( m = \text{rank} F_M \).

**Proof.** Let \( r \) be a nonnegative integer. The matrix \( F_{m+r} \) can be written as

\[
F_{m+r} = \sum_{k=1}^{m} \lambda_k \begin{bmatrix}
\psi_0 (x_k) \psi_0 (x_k) & \cdots & \psi_0 (x_k) \psi_{m+r-1} (x_k) \\
\vdots & \ddots & \vdots \\
\psi_{m+r-1} (x_k) \psi_0 (x_k) & \cdots & \psi_{m+r-1} (x_k) \psi_{m+r-1} (x_k)
\end{bmatrix}
\]

\[
= \sum_{k=1}^{m} \lambda_k \begin{bmatrix}
\psi_0 (x_k) \\
\vdots \\
\psi_{m+r-1} (x_k)
\end{bmatrix}
\begin{bmatrix}
\psi_0 (x_k) & \cdots & \psi_{m+r-1} (x_k)
\end{bmatrix}.
\]

This implies that \( \text{rank} F_{m+r} \leq m \). However, \( F_m \) is nonsingular. Indeed, one can easily verify that \( F_m \) can be factorized as \( F_m = V_m D_m V_m^T \) where \( V_m \) is the Vandermonde-like matrix

\[
V_m := \left[ \psi_r (x_s) \right]_{r=0,s=1}^{m-1,m}
\]

and \( D_m \) is the diagonal matrix

\[
D_m := \text{diag}(\lambda_1, \ldots, \lambda_m).
\]

Therefore \( \text{rank} F_{m+r} \geq m \). It follows that \( \text{rank} F_{m+r} = m \).

Thus the regular FOP of degree \( m \) exists whereas regular FOPs of degree \( > m \) do not exist. Note that, in contrast to the setting of Chapters 1 and 2, the regular FOP of degree one need not exist. This will be the only important point in which the algorithm that we will present will be different from the algorithm that we have presented in Chapter 1. If \( \langle 1, 1 \rangle = \sum_{k=1}^{m} \lambda_k \neq 0 \), then the regular FOP of degree
one exists and it is given by \( \varphi_1(z) = z - \mu \) where \( \mu := \langle 1, z \rangle / \langle 1, 1 \rangle \). In the case of meromorphic functions, the condition \( \langle 1, 1 \rangle \neq 0 \) means that \( N \neq P \). For analytic functions, we have that \( P = 0 \) and thus the regular FOP of degree one always exists (if we assume, of course, that \( N > 0 \)).

The following theorem shows how \( x_1, \ldots, x_m \) can be computed by solving a generalized eigenvalue problem.

**Theorem 36.** The eigenvalues of the pencil \( F_m^{(1)} - \lambda F_m \) are \( \psi_1(x_1), \ldots, \psi_1(x_m) \).

**Proof.** Define \( V_m \) as the Vandermonde-like matrix
\[
V_m := [\psi_r(x_i)]_{r=0, i=1}^{m-1, m}
\]
and let \( D_m \) and \( D_m^{(1)} \) be the diagonal matrices
\[
D_m := \text{diag}(\lambda_1, \ldots, \lambda_m) \quad \text{and} \quad D_m^{(1)} := \text{diag}(\lambda_1 \psi_1(x_1), \ldots, \lambda_m \psi_1(x_m)).
\]
Then \( F_m \) and \( F_m^{(1)} \) can be factorized as \( F_m = V_mD_m V_m^T \) and \( F_m^{(1)} = V_mD_m^{(1)} V_m^T \).

Let \( \lambda^* \) be an eigenvalue of the pencil \( F_m^{(1)} - \lambda F_m \) and \( x \) a corresponding eigenvector. Then
\[
F_m^{(1)} x = \lambda^* F_m x \\
\Leftrightarrow (V_mD_m^{(1)} V_m^T)x = \lambda^*(V_mD_m V_m^T)x \\
\Leftrightarrow D_m^{(1)} y = \lambda^* D_m y \quad \text{if} \quad y := V_m x \\
\Leftrightarrow \text{diag}(\psi_1(x_1), \ldots, \psi_1(x_m)) y = \lambda^* y.
\]
This proves the theorem. \( \square \)

As \( \psi_1(z) \) is a polynomial of degree one, the “nodes” \( x_1, \ldots, x_m \) can be calculated by applying the previous theorem. Once \( m \) and \( x_1, \ldots, x_m \) have been found, the problem becomes linear and the “weights” \( \lambda_1, \ldots, \lambda_m \) can be computed by solving a Vandermonde-like system of linear equations.

The question remains, of course, which polynomials \( \psi_k(z) \) to choose. Again, as in Chapter 1, we will obtain very accurate numerical results by using the formal orthogonal polynomials associated with \( \langle \cdot, \cdot \rangle \).

If \( F_m \) is strongly nonsingular, then we have a full set \( \{ \varphi_0, \varphi_1, \ldots, \varphi_m \} \) of regular FOPs. If not, then the gaps in the sequence of existing regular FOPs can be filled up in the same way as before. We may therefore assume that a suitable sequence \( \{ \varphi_k \}_{k=0}^\infty \) has been defined.

It is a trivial task to generalize Theorems 7 (concerning the computation of zeros of regular FOPs) and 9 (concerning the stopping criterion/the determination of the value of \( m \)).

These considerations lead to the following algorithm.

**Algorithm**

**input** \( \langle \cdot, \cdot \rangle, M, \epsilon_{\text{stop}} \)

**output** \( m, \) nodes

**comment** nodes = \( \{x_1, \ldots, x_m\} \). We assume that \( M \geq m \) and \( \epsilon_{\text{stop}} > 0 \).

\( \varphi_0(z) \leftarrow 1 \)
\[
\begin{align*}
r &\leftarrow 0 \\
s_0 &\leftarrow \{1,1\} \\
\text{if } s_0 &= 0 \text{ then} \\
\mu &\leftarrow 0 \\
\varphi_1(z) &\leftarrow z \\
t &\leftarrow 1 \\
\text{else} \\
\mu &\leftarrow \{1, z\}/s_0 \\
\varphi_1(z) &\leftarrow z - \mu \\
r &\leftarrow 1; t &\leftarrow 0 \\
\text{end if} \\
\text{while } r + t < M \text{ do} \\
\text{regular } &\leftarrow \text{it is numerically feasible to generate } \varphi_{r+t+1}(z) \text{ as} \\
&\text{a regular FOP} \\
\text{if } \text{regular} \text{ then} \\
\text{generate } \varphi_{r+t+1}(z) \text{ as a regular FOP} \\
r &\leftarrow r + t + 1; t &\leftarrow 0 \\
\text{allsmall } &\leftarrow \text{true}; \tau &\leftarrow 0 \\
\text{while } \text{allsmall and } (r + \tau < M) \text{ do} \\
[i_p, \text{maxpsum}] &\leftarrow ((z - \mu)^T \varphi_r(z), \varphi_r(z)) \\
i_p &\leftarrow [i_p] \\
\text{allsmall } &\leftarrow (i_p/\text{maxpsum} < \epsilon_{\text{stop}}) \\
\tau &\leftarrow \tau + 1 \\
\text{end while} \\
\text{if } \text{allsmall } \text{ then} \\
m &\leftarrow r; \text{nodes } &\leftarrow \text{roots}(\varphi_r); \text{stop} \\
\text{end if} \\
\text{else} \\
generate \varphi_{r+t+1}(z) \text{ as an inner polynomial} \\
t &\leftarrow t + 1 \\
\text{end if} \\
\text{end while} \\
m &\leftarrow M; \text{nodes } &\leftarrow \text{roots}(\varphi_M); \text{stop}
\end{align*}
\]

Similar comments as those given after the formulation of the algorithm in Chapter 1 apply.

3. A numerical example

We have modified the Matlab implementation that we have already used in Section 4 of Chapter 1.

**Example 18.** Suppose that
\[
f(z) = \frac{1}{z^2(z - 1)(z^2 + 9)} + z \sin z + e^{-3z} + 4
\]
and let \( \gamma \) be the circle \( \{z \in \mathbb{C} : |z| = 2\} \). We set the upper bound \( \hat{P} = 5 \). It turns out that \( \{1,1\} = 0 \) (in other words, \( N = P \)) and thus \( M = s_0 + 2\hat{P} = 10 \). We set
Our algorithm reacts as follows. It defines \( \varphi_0(z) \) as a regular FOP, \( \varphi_1(z) \) as an inner polynomial and \( \varphi_2(z) \) as a regular FOP. The scaled counterpart of \( |(\varphi_2(z), \varphi_2(z))| \) is equal to

\[
4.821901380151357 \times 10^{-8}
\]

The algorithm defines \( \varphi_3(z) \) as an inner polynomial and \( \varphi_4(z) \) as a regular FOP. The scaled counterpart of \( |(\varphi_4(z), \varphi_4(z))| \) is equal to

\[
1.77652453847086 \times 10^{-1}
\]

The algorithm defines \( \varphi_5(z) \) as a regular FOP. For \( k = 0, 1, \ldots, 4 \), the scaled counterparts of \( |z^k \varphi_5(z), \varphi_5(z)| \) are given by

\[
\begin{align*}
&2.387781621321191 \times 10^{-15} \\
&2.887901536460305 \times 10^{-15} \\
&1.26859438890019 \times 10^{-15} \\
&3.904276363179922 \times 10^{-15} \\
&2.77260727600076 \times 10^{-15}
\end{align*}
\]

The algorithm decides that \( n + p = 5 \) and stops. The absolute errors of the computed approximations for the zeros and poles are \( O(10^{-12}) \). After one step of iterative refinement, we obtain the following results:

<table>
<thead>
<tr>
<th>( x_k )</th>
<th>( \lambda_k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.97843635600921</td>
<td>1</td>
</tr>
<tr>
<td>0.16974891913248</td>
<td>1</td>
</tr>
<tr>
<td>-0.13327146070751</td>
<td>1</td>
</tr>
<tr>
<td>1.00000000000000</td>
<td>-1</td>
</tr>
<tr>
<td>0.00000000000000</td>
<td>-2</td>
</tr>
</tbody>
</table>

Note how zeros and poles can be distinguished by checking the signs of the \( \lambda_k \)'s. ※

**Note.** As our algorithm provides not only approximations for the zeros and poles but also the corresponding multiplicities and orders, we can use the modified Newton's method

\[
x_k^{(\alpha+1)} = x_k^{(\alpha)} - \nu_k \frac{f(z_k^{(\alpha)})}{f'(z_k^{(\alpha)})}, \quad y_l^{(\alpha+1)} = y_l^{(\alpha)} + \mu_l \frac{f(y_l^{(\alpha)})}{f'(y_l^{(\alpha)})}, \quad \alpha = 0, 1, 2, \ldots
\]

to refine the approximations for the zeros and poles.
CHAPTER 4

Systems of analytic equations

In Chapter 1 we have considered the problem of computing all the zeros of an analytic function \( f \) that lie in the interior of a Jordan curve \( \gamma \). The algorithm that we have presented computes not only accurate approximations for the zeros but also their respective multiplicities. It does not require initial approximations for the zeros. In Chapter 2 we have seen how our algorithm can be used to locate clusters of zeros of analytic functions whereas in Chapter 3 we have adapted it to handle the problem of calculating zeros and poles of meromorphic functions. As our approach relies upon integrals along \( \gamma \) that involve the logarithmic derivative \( f' / f \), it could be called a logarithmic residue based approach.

In this chapter we will present a logarithmic residue based approach for the problem of computing zeros of analytic mappings (in other words, for solving systems of analytic equations). A multidimensional logarithmic residue formula is available in the literature. This formula involves the integral of a differential form, which we will transform into a sum of Riemann integrals. More precisely, if \( d \) denotes the dimension of the mapping (i.e., the number of equations, which is assumed to be equal to the number of variables), then this sum consists of \( d \) Riemann integrals of dimension \( 2d - 1 \). We will show how the zeros and their respective multiplicities can be computed from these integrals by solving a generalized eigenvalue problem that has Hankel structure and \( d \) Vandermonde systems, cf. Theorems 3 and 4 and Equation (30).

It turns out that these integrals are difficult to evaluate numerically. Therefore we prefer to use ordinary moments instead of modified moments, even if this means that the computed approximations for the zeros will be less accurate. By using a cubature package that can handle vectors of similar integrals over a common integration region, we will be able to calculate all the ordinary moments that are needed simultaneously. This will enable us to reduce the cost of our approach. Indeed, a significant part of the computation required for each integrand will be the same for all of the integrands and thus these common calculations need to be done only once for each integrand evaluation point.

This chapter corresponds to our paper [183].

1. Introduction

Let \( d \geq 1 \) be a positive integer. Consider a polydisk \( D \) in \( \mathbb{C}^d \) (i.e., \( D \) is the Cartesian product of \( d \) disks in \( \mathbb{C} \)) and let \( f = (f_1, \ldots, f_d) : \mathbb{T} \to \mathbb{C}^d \) be a mapping that is analytic in \( \mathbb{T} \) and has no zeros on the boundary of \( D \). The latter implies that \( f \) has only a finite number of zeros in \( D \) and that these zeros are all isolated [3,
Theorem 2.4]. We consider the problem of computing these zeros, together with their respective multiplicities. (For a precise definition of the multiplicity of a zero of an analytic mapping, we refer to Chapter 7.)

Let $Z_f(D)$ denote the set of zeros of $f$ that lie in $D$ and let $\mu_a(f)$ denote the multiplicity of a zero $a \in Z_f(D)$.

In case $d = 1$, the classical logarithmic residue formula that we have used in the previous chapters tells us that

$$\frac{1}{2\pi i} \int_{\partial D} \varphi(z) \frac{f'(z)}{f(z)} dz = \sum_{a \in Z_f(D)} \mu_a(f) \varphi(a)$$

if $\varphi : D \to \mathbb{C}$ is analytic in $D$ and continuous in $\overline{D}$. By evaluating the integral in the left-hand side numerically, we have been able to obtain information about the location of the zeros of $f$.

A multidimensional generalization of (48) is available in the theory of functions of several complex variables [3, Theorem 3.1]. It involves the integral of a differential form. To prepare for the numerical evaluation of this integral, we transform it into a Riemann integral, or rather, a sum of $d$ Riemann integrals. This result is formulated in Theorem 37 and looks as follows:

$$I(\varphi) := \sum_{k=1}^d I_k(\varphi) = \sum_{a \in Z_f(D)} \mu_a(f) \varphi(a)$$

if $\varphi : \mathcal{D} \to \mathbb{C}$ is analytic in $D$ and continuous in $\overline{D}$ and where $I_1(\varphi), \ldots, I_d(\varphi)$ are certain Riemann integrals over the unit cube in $\mathbb{R}^{d-1}$. Observe that the total number of zeros of $f$ that lie in $D$ is equal to $I(1)$.

The proof of Theorem 37 is formulated in the language of differential forms. This cannot be avoided. Readers who feel less at ease with differential forms may consult [62], [83] or [287] for an introduction and [222] for a thorough exposition of analysis on complex manifolds.

Unfortunately, the integrals that appear in Theorem 37 tend to be difficult to evaluate numerically. The efficient numerical evaluation of these integrals is still an open problem. It represents a challenge for the numerical integration community. (More details about the various numerical integration algorithms that we have tried will be given in Section 4.)

Note. These integrals are similar to the Kronecker-Picard integrals that appear in topological degree based methods for computing solutions to twice continuously-differentiable systems of real equations. See, for example, Erdelsky [76], O’Neil and Thomas [228] and also Ragoš, Vrahatis and Zafiropoulos [243], and Kavvadias and Vrahatis [172].

We will therefore assume that we are able to evaluate the functional $I(\varphi)$ for every function $\varphi$ that satisfies the hypotheses of Theorem 37. In particular, we will suppose that the total number of zeros of $f$ that lie in $D$ can be computed. Our unknowns are the number of mutually distinct zeros of $f$ that lie in $D$, these zeros themselves, and their respective multiplicities. In Section 3 we will show how specific
choices of $\varphi$ enable us to calculate our unknowns by solving a generalized eigenvalue problem that has Hankel structure and $d$ Vandermonde systems.

Note. Algebraic mappings are of course a special case of analytic mappings. Systems of polynomial equations have received considerable interest in recent years. Several classes of methods have been developed for their solution: Gröbner bases, homotopy continuation, sparse resultants and interval methods (see, for example, [53, 74, 78, 200, 218, 282, 286]). We will not compare our approach with these methods because they have been developed specifically for systems of polynomial equations whereas we consider systems of arbitrary analytic equations, a problem that has received much less attention in the literature.

2. A multidimensional logarithmic residue formula

Let $J_f$ denote the Jacobian matrix of $f$ and let $J_{[k]}$ be the Jacobian matrix of $f$ with the $k$th column replaced with $[f_1 \cdots f_d]^T$:

$$J_{[k]} := \begin{bmatrix}
\frac{\partial f_1}{\partial z_1} & \cdots & \frac{\partial f_1}{\partial z_d} & f_1 \\
\vdots & & \vdots & \vdots \\
\frac{\partial f_d}{\partial z_1} & \cdots & \frac{\partial f_d}{\partial z_d} & f_d
\end{bmatrix}, \quad k = 1, \ldots, d.$$ 

Suppose also that the polydisk $D$ is given by

$$D = D_1 \times \cdots \times D_d$$

where

$$D_k = \{ z \in \mathbb{C} : |z - C_k| < R_k \}, \quad k = 1, \ldots, d,$$

with $C_1, \ldots, C_d \in \mathbb{C}$ and $R_1, \ldots, R_d > 0$.

Theorem 37. Let $\varphi : \overline{D} \to \mathbb{C}$ be analytic in $D$ and continuous in $\overline{D}$. Define $I_k(\varphi)$ for $k = 1, \ldots, d$ as the integral

$$I_k(\varphi) := \rho_k \int_{p,1}^{p,1} \frac{\varphi(z_1, \ldots, z_d) \det J_f(z_1, \ldots, z_d) \det J_{[k]}(z_1, \ldots, z_d)}{(|f_1(z_1, \ldots, z_d)|^2 + \cdots + |f_d(z_1, \ldots, z_d)|^2)^d}$$

$$\times e^{2\pi i \theta_z r_1} \cdots r_{k-1} r_{k+1} \cdots r_d \, dr_1 \cdots dr_{k-1} \, dr_{k+1} \cdots dr_d \, d\theta_1 \cdots d\theta_d$$

with

$$\rho_k = \rho(d, R_1, \ldots, R_d; k) := 2^{d-1} (d-1)! \frac{R_1^2 \cdots R_{k-1}^2 R_k R_{k+1}^2 \cdots R_d^2}{(\Re \varphi)^2}$$

and where

$$z_k = z_k(\theta_k) = C_k + R_k e^{2\pi i \theta_k} \quad 0 \leq \theta_k \leq 1$$

and

$$z_l = z_l(\theta_l) = C_l + R_l e^{2\pi i \theta_l} \quad 0 \leq \theta_l \leq 1$$

for $l \in \{1, \ldots, d\} \setminus \{k\}$. Then

$$I(\varphi) := \sum_{k=1}^d I_k(\varphi) = \sum_{a \in Z_1(D)} \mu_a(f) \varphi(a).$$

Theorem 37 is a corollary of the following theorem.
Theorem 38. Let $D$ be a polydisk in $\mathbb{C}^d$ and let $f = (f_1, \ldots, f_d): \overline{D} \to \mathbb{C}$ be a mapping that is analytic in $\overline{D}$ and has no zeros on the boundary of $D$. Let $\varphi: \overline{D} \to \mathbb{C}$ be analytic in $D$ and continuous in $\partial D$. Define the differential form $\omega(f, \overline{f})$ as

$$\omega(f, \overline{f}) := \frac{(d - 1)!}{(2\pi i)^d} \frac{1}{|f|^2} \sum_{k=1}^{d} (-1)^{k-1} \overline{f}_k \, df_k \wedge df.$$

Then

$$\int_{\partial D} \varphi \omega(f, \overline{f}) = \sum_{a \in Z_f(D)} \mu_a(f) \varphi(a).$$

Proof. Yuzhakov and Roos [3, Theorem 3.1] proved this result for arbitrary bounded domains in $\mathbb{C}^d$ with a piecewise smooth boundary.

The notations used in the formulation of the previous theorem,

$$\overline{f} = (\overline{f_1}, \ldots, \overline{f_d})$$

$$|f| = \sqrt{|f_1|^2 + \cdots + |f_d|^2}$$

$$df = df_1 \wedge \cdots \wedge df_d$$

$$df_k = df_1 \wedge \cdots \wedge \overline{df_k} \wedge \cdots \wedge df_d, \quad k = 1, \ldots, d,$$

are classical. (The tilde over the form $df_k$ means that this form is omitted and does not appear in the product.)

To get rid of the differential forms in Theorem 38, we proceed as follows. Define the form $\eta(f)$ as

$$\eta(f) := \sum_{k=1}^{d} (-1)^{k-1} f_k \, df_1 \wedge \cdots \wedge \overline{df_k} \wedge \cdots \wedge df_d.$$

This form is sometimes called the Leray form [182]. Then

$$\omega(f, \overline{f}) = \frac{(d - 1)!}{(2\pi i)^d} \frac{1}{|f|^2} \eta(f) \wedge df.$$

If $f = (z_1, \ldots, z_d)$, then $df = \det J_f(z_1, \ldots, z_d) \, dz_1 \wedge \cdots \wedge dz_d$. The following lemma shows what happens with the Leray form in this case.

Lemma 39. If $f = (z_1, \ldots, z_d)$, then

$$\eta(f) = \sum_{k=1}^{d} (-1)^{k-1} \det J_{[k]}(z_1, \ldots, z_d) \, dz_1 \wedge \cdots \wedge \overline{dz_k} \wedge \cdots \wedge dz_d.$$

Proof. Let $j$ be an integer between 1 and $d$, $j \in \{1, \ldots, d\}$. Then

$$\sum_{k=1}^{d} (-1)^{k-1} \det J_{[k]}(dz_1 \wedge \cdots \wedge \overline{dz_k} \wedge \cdots \wedge dz_d)(\frac{\partial}{\partial z_1}, \ldots, \frac{\partial}{\partial z_j}, \ldots, \frac{\partial}{\partial z_d})$$

$$= (-1)^{j-1} \det J_{[j]}.$$
By expanding $\det J_{[j]}$ along the $j$th column, we obtain that

$$(-1)^{j-1} \det J_{[j]} = (-1)^{j-1} \sum_{k=1}^{d} (-1)^{k+j} f_k \det \begin{bmatrix} \frac{\partial f_1}{\partial z_1} & \cdots & \frac{\partial f_1}{\partial z_{j-1}} & \frac{\partial f_1}{\partial z_{j+1}} & \cdots & \frac{\partial f_1}{\partial z_d} \\ \vdots & & \vdots & \vdots & & \vdots \\ \frac{\partial f_{k-1}}{\partial z_1} & \cdots & \frac{\partial f_{k-1}}{\partial z_{j-1}} & \frac{\partial f_{k-1}}{\partial z_{j+1}} & \cdots & \frac{\partial f_{k-1}}{\partial z_d} \\ \frac{\partial f_k}{\partial z_1} & \cdots & \frac{\partial f_k}{\partial z_{j-1}} & \frac{\partial f_k}{\partial z_{j+1}} & \cdots & \frac{\partial f_k}{\partial z_d} \\ \vdots & & \vdots & \vdots & & \vdots \\ \frac{\partial f_d}{\partial z_1} & \cdots & \frac{\partial f_d}{\partial z_{j-1}} & \frac{\partial f_d}{\partial z_{j+1}} & \cdots & \frac{\partial f_d}{\partial z_d} \end{bmatrix}.$$  

The determinant in the right-hand side is by definition equal to

$$(df_1 \wedge \cdots \wedge \widetilde{df_k} \wedge \cdots \wedge df_d)(\frac{\partial}{\partial z_1}, \ldots, \frac{\partial}{\partial z_j}, \ldots, \frac{\partial}{\partial z_d})$$

This proves the lemma.  

It follows that

$$\omega(f, \overline{f}) = \sum_{k=1}^{d} \omega_k(f, \overline{f})$$

where

$$\omega_k(f, \overline{f}) := (-1)^{k-1} \frac{(d-1)!}{(2\pi i)^d} \frac{\det J_f \det J_{[k]}}{|J_f|^d} dz_k \wedge dz, \quad k = 1, \ldots, d.$$ 

The boundary of $D$ is given by

$$\partial D = \partial D_{[1]} \cup \cdots \cup \partial D_{[d]}$$

where

$$\partial D_{[k]} := D_1 \times \cdots \times D_{k-1} \times \partial D_k \times D_{k+1} \times \cdots \times D_d, \quad k = 1, \ldots, d,$$

and thus

$$\int_{\partial D} \varphi \omega(f, \overline{f}) = \sum_{k=1}^{d} \int_{\partial D_{[k]}} \varphi \omega_k(f, \overline{f}).$$

Now let us introduce polar coordinates. Let $C \in \mathbb{C}$ and $R > 0$. If $z = z(\theta) = C + Re^{2\pi i \theta}$, then $dz = 2\pi i Re^{2\pi i \theta} d\theta$. And if $z = z(r, \theta) = C + r Re^{2\pi i \theta}$, then $dz = (2\pi i)rR^2 \, dr \wedge d\theta$. Thus

$$\int_{\partial D_{[k]}} \varphi \omega_k(f, \overline{f})$$

$$= (-1)^{k-1} \rho_k \int_{\partial D_{[k]}} \frac{\varphi \det J_f \det J_{[k]}}{(|f_1|^2 + \cdots + |f_d|^2)^d} \left[ \prod_{l=1, l \neq k}^d r_l \right] e^{2\pi i \theta k} \, dr \wedge d\theta$$

for $k = 1, \ldots, d$, where

$$\rho_k = \rho(d, R_1, \ldots, R_d; k) := 2^{d-1}(d-1)! R_1^{2k} \cdots R_{k-1}^{2k} R_k R_{k+1}^2 \cdots R_d^2.$$
and in the integral on the right-hand side
\[ z_k = z_k(\theta_k) = C_k + R_k e^{2\pi i \theta_k} \]
and
\[ z_l = z_l(r_l, \theta_l) = C_l + r_l R_l e^{2\pi i \theta_l}, \]
for \( l \in \{1, \ldots, d\} \setminus \{k\} \). In [3] it is assumed that \( \mathbb{C}^d \) has the orientation determined by the form \( dr_1 \wedge \cdots \wedge dr_d \wedge d\theta_1 \wedge \cdots \wedge d\theta_d \) and that the boundary of \( D \) is assigned the orientation induced by that of \( D \). This implies that \( \partial D_{[a]} \) has the orientation determined by the form \((-)^{k-1} dr_{[a]} \wedge d\theta \). Therefore
\[
\int_{\partial D_{[a]}} \varphi \omega_k(f, \overline{T}) = \rho_k \int_{[0,1]^{d-1}} \frac{\varphi(z_1, \ldots, z_d) \det J_f(z_1, \ldots, z_d) \det J_{[a]}(z_1, \ldots, z_d)}{|f_1(z_1, \ldots, z_d)|^2 + \cdots + |f_d(z_1, \ldots, z_d)|^2} \times e^{2\pi i \theta_1} r_1 \cdots r_{k-1} r_{k+1} \cdots r_d \, dr_1 \cdots dr_{k-1} \cdots dr_{k+1} \cdots dr_d \, d\theta_1 \cdots d\theta_d
\]
for \( k = 1, \ldots, d \), where
\[ z_k = z_k(\theta_k) = C_k + R_k e^{2\pi i \theta_k} \quad 0 \leq \theta_k \leq 1 \]
and
\[ z_l = z_l(r_l, \theta_l) = C_l + r_l R_l e^{2\pi i \theta_l} \quad 0 \leq r_l, \theta_l \leq 1 \]
for \( l \in \{1, \ldots, d\} \setminus \{k\} \). This proves Theorem 37.

3. The algorithm

In this section we will show how the zeros and their respective multiplicities can be computed by solving a generalized eigenvalue problem that has Hankel structure and d Vandermonde systems. Our approach is inspired by Theorems 3 and 4 and Equation (30). We have already explained why we prefer to use ordinary moments instead of modified moments.

First we introduce some notation. The total number of zeros of \( f \) that lie in \( D \) will be denoted by \( N \). As explained in Section 1 we will assume that the value of \( N \) can be computed numerically. From now on, we will also suppose that \( N > 0 \). Let \( n \) be the number of mutually distinct zeros of \( f \) that lie in \( D \). Let
\[
(z_1^{(1)}, \ldots, z_d^{(1)}), \ldots, (z_1^{(n)}, \ldots, z_d^{(n)})
\]
denote these zeros and let \( \nu_1, \ldots, \nu_n \) be their respective multiplicities. Without loss of generality we may assume that \( z_d^{(p)} \neq z_d^{(q)} \), \( p \neq q \). Indeed, if one first applies a random unitary linear transformation to the unknowns \( z = (z_1, \ldots, z_d) \), then this condition is satisfied almost surely, i.e., with probability one. Analogous results can be formulated in case \( z_k^{(p)} \neq z_k^{(q)} \), \( p \neq q \), for some \( k \in \{1, \ldots, d-1\} \). We leave this to the reader. What happens if our algorithm is applied in case the \( d \)th components of the zeros of \( f \) that lie in \( D \) are not mutually distinct, will be discussed at the end of this section.

Define \( s_p \) for \( p = 0, 1, 2, \ldots \) as
\[ s_p := I(z_d^p) \]
where \( I(\cdot) \) is defined in (49). We will assume that the sequence \((s_p)_{p \geq 0}\) can be computed numerically. By Theorem 37

\[
s_p = \nu_1 [z_d^{(1)}]^p + \cdots + \nu_n [z_d^{(n)}]^p, \quad p = 0, 1, 2, \ldots
\]

In particular, \( s_0 = N \). Define

\[
H_k := \begin{bmatrix}
s_0 & s_1 & \cdots & s_{k-1} \\
s_1 & \ddots & \ddots & \vdots \\
\vdots & \ddots & \ddots & \vdots \\
s_{k-1} & \cdots & \cdots & s_{2k-2}
\end{bmatrix}
\]

for \( k = 1, 2, \ldots \). Also, let \( V_n \) be the Vandermonde matrix with nodes \( z_1^{(1)}, \ldots, z_n^{(n)} \),

\[
V_n := \begin{bmatrix}
[z_d^{(1)}]_{k=1}^n \\
[z_d^{(2)}]_{k=1}^n \\
\vdots \\
[z_d^{(n)}]_{k=1}^n
\end{bmatrix}
\]

The following theorem characterizes \( n \), the number of mutually distinct zeros, cf. Theorem 3.

**Theorem 40.** \( n = \text{rank} H_{n+p} \) for every nonnegative integer \( p \). In particular, \( n = \text{rank} H_N \).

**Proof.** The proof is similar to that of Theorem 3. It is based on a factorization of the matrix \( H_n \). The Vandermonde matrix \( V_n \) is regular since \( z_d^{(1)}, \ldots, z_d^{(n)} \) are assumed to be mutually distinct.

The \( d \)th components \( z_1^{(1)}, \ldots, z_n^{(n)} \) of the zeros can be calculated by solving a generalized eigenvalue problem that has Hankel structure, cf. Theorem 4. Define

\[
H_n^< := [s_1 + k + 1]_{k,l=0}^{n-1}
\]

**Theorem 41.** The eigenvalues of the pencil \( H_n^< - \lambda H_n \) are given by \( z_1^{(1)}, \ldots, z_n^{(n)} \).

**Proof.** The proof is similar to that of Theorem 4.

Once \( z_1^{(1)}, \ldots, z_n^{(n)} \) have been found, the multiplicities \( \nu_1, \ldots, \nu_n \) can be computed by solving the Vandermonde system

\[
V_n \begin{bmatrix}
\nu_1 \\
\nu_2 \\
\vdots \\
\nu_n
\end{bmatrix} = \begin{bmatrix}
s_0 \\
s_1 \\
\vdots \\
s_{n-1}
\end{bmatrix}
\]

**Remark.** Theoretically the \( N - n \) smallest singular values of \( H_N \) are equal to zero. In practice, as by evaluating the corresponding integrals numerically we can only obtain approximations for the \( s_p \)'s and because of roundoff errors in the SVD computation, this will not be the case. If the numerical rank \( n \) of \( H_N \) is difficult to determine, it is safe to consider \( H_N \) as a matrix of full rank and to solve an \( N \times N \) generalized eigenvalue problem and associated Vandermonde system.
Theorem 42. For every integer \( \alpha \geq n \) the eigenvalues of the pencil
\[
[s_{1+k+l}]_{k,l=0}^{\alpha-1} - \lambda [s_{k+l}]_{k,l=0}^{\alpha-1}
\]
are given by \( z_d^{(1)}, \ldots, z_d^{(n)} \) and \( \alpha - n \) eigenvalues that may assume arbitrary values.

Proof. This follows from Theorem 41 and by taking into account that \((s_p)_{p \geq 0}\)
is a linear recurring sequence:
\[
s_{n+p} + \tau_1 s_{n-1+p} + \cdots + \tau_n s_p = 0, \quad p = 0, 1, 2, \ldots,
\]
if \( \tau_1, \ldots, \tau_n \) are defined as the coefficients of the monic polynomial of degree \( n \) that
has \( z_d^{(1)}, \ldots, z_d^{(n)} \) as simple zeros,
\[
\prod_{k=1}^{n} (z - z_d^{(k)}) =: z^n + \tau_1 z^{n-1} + \cdots + \tau_n.
\]
Note that the latter implies that the matrices
\[
[s_{1+k+l}]_{k,l=0}^{\alpha-1} \quad \text{and} \quad [s_{k+l}]_{k,l=0}^{\alpha-1}
\]
have the same null spaces. \( \square \)

The previous theorem tells us that the generalized eigenvalue problem (51) has \( \alpha - n \)
eigenvalues that may assume arbitrary values. Each of these indeterminate eigenvalues corresponds to two corresponding zeros on the diagonals of the generalized Schur decomposition of the Hankel matrices \([s_{1+k+l}]_{k,l=0}^{\alpha-1}\) and \([s_{k+l}]_{k,l=0}^{\alpha-1}\). When actually calculated, these diagonal entries are different from zero because of roundoff errors. The quotient of two such corresponding diagonal entries is an eigenvalue that
is not the \( d \)th component of a zero of \( f \). Fortunately, cf. the concluding paragraph of
Example 4, the corresponding Vandermonde system enables us to detect such spurious
\( d \)th components. Indeed, the Vandermonde matrix whose nodes are given by
the eigenvalues of (51) is still regular and therefore the corresponding Vandermonde
system has only one solution, which gives the true \( d \)th components their correct
multiplicity and the spurious ones “multiplicity” zero.

Once \( n, z_d^{(1)}, \ldots, z_d^{(n)} \) and \( \nu_1, \ldots, \nu_n \) have been found, the unknowns
\[
z_1^{(1)}, \ldots, z_1^{(n)}, \ldots, z_{d-1}^{(1)}, \ldots, z_{d-1}^{(n)}
\]
can be obtained as follows. Define \( t_{k, p} \) for \( k = 1, \ldots, d - 1 \) and \( p = 0, 1, 2, \ldots \) as
\[
t_{k, p} := I(z_k z_d^{(p)}).
\]
We will assume that the sequences \((t_{k, p})_{p \geq 0}\) can be computed for \( k = 1, \ldots, d - 1 \).
By Theorem 37
\[
t_{k, p} = \nu_k z_k^{(1)} [z_d^{(1)}]^p + \cdots + \nu_k z_k^{(n)} [z_d^{(n)}]^p
\]
for \( k = 1, \ldots, d-1 \) and \( p = 0, 1, 2, \ldots \). It follows that \( z_k^{(1)}, \ldots, z_k^{(n)} \) can be calculated from the solution of the Vandermonde system

\[
V_n \begin{bmatrix}
\nu_1 z_k^{(1)} \\
\nu_2 z_k^{(2)} \\
\vdots \\
\nu_n z_k^{(n)}
\end{bmatrix} = \begin{bmatrix}
t_{k,0} \\
t_{k,1} \\
\vdots \\
t_{k,n-1}
\end{bmatrix}, \quad k = 1, \ldots, d-1.
\]

As already discussed in Chapter 1, problems of numerical linear algebra that involve Vandermonde or Hankel matrices truly deserve their reputation of being ill-conditioned \([98, 271]\). By moving the origin in the \( z_i \)-plane to the arithmetic mean of the \( d \)th components of the zeros,

\[
z_d' := \frac{\nu_1 z_d^{(1)} + \cdots + \nu_n z_d^{(n)}}{\nu_1 + \cdots + \nu_n} = \frac{I(z_d)}{I(1)},
\]

ill-conditioning is reduced significantly. Therefore we will use shifted versions of the integrals \( s_p \) and \( t_{k,p} \), denoted by \( \hat{s}_p \) and \( \hat{t}_{k,p} \). The results of this section then lead to the following algorithm.

**Algorithm**

1. \( N \leftarrow I(1) \)
2. \( \hat{z}_d \leftarrow I(z_d)/N \)
3. \( \hat{s}_0 \leftarrow N; \quad \hat{s}_1 \leftarrow 0; \quad \hat{s}_p \leftarrow I((z_d - z_d')^p) \) for \( p = 2, \ldots, 2N-1 \)
4. \( n \leftarrow \text{rank} [\hat{s}_{k+l}]_{l=0}^{N-1} \)
5. Calculate the eigenvalues \( \lambda_1, \ldots, \lambda_n \) of the pencil \([\hat{s}_{1+k+l}]_{l=0}^{n-1} - \lambda [\hat{s}_{k+l}]_{l=0}^{n-1}\).
6. \( z_d^{(k)} \leftarrow \lambda_k + \hat{s}_d \) for \( k = 1, \ldots, n \)
7. Solve the Vandermonde system

\[
\begin{bmatrix}
1 & 1 & \cdots & 1 \\
\lambda_1 & \lambda_2 & \cdots & \lambda_n \\
\vdots & \vdots & \ddots & \vdots \\
\lambda_1^{n-1} & \lambda_2^{n-1} & \cdots & \lambda_n^{n-1}
\end{bmatrix}
\begin{bmatrix}
\nu_1 \\
\nu_2 \\
\vdots \\
\nu_n
\end{bmatrix}
= \begin{bmatrix}
\hat{s}_0 \\
\hat{s}_1 \\
\vdots \\
\hat{s}_{n-1}
\end{bmatrix}.
\]
8. \( \hat{t}_{k,p} \leftarrow I((z_d(z_d - z_d'))^p) \) for \( k = 1, \ldots, d-1 \) and \( p = 0, \ldots, n-1 \)
9. Solve the Vandermonde system

\[
\begin{bmatrix}
1 & 1 & \cdots & 1 \\
\lambda_1 & \lambda_2 & \cdots & \lambda_n \\
\vdots & \vdots & \ddots & \vdots \\
\lambda_1^{n-1} & \lambda_2^{n-1} & \cdots & \lambda_n^{n-1}
\end{bmatrix}
\begin{bmatrix}
x_{1,1} & \cdots & x_{d-1,1} \\
x_{1,2} & \cdots & x_{d-1,2} \\
\vdots & \vdots & \ddots \\
x_{1,n} & \cdots & x_{d-1,n}
\end{bmatrix}
= \begin{bmatrix}
\hat{t}_{1,0} & \cdots & \hat{t}_{d-1,0} \\
\hat{t}_{1,1} & \cdots & \hat{t}_{d-1,1} \\
\vdots & \vdots & \ddots \\
\hat{t}_{1,n-1} & \cdots & \hat{t}_{d-1,n-1}
\end{bmatrix}.
\]

\( z_k^{(l)} \leftarrow x_{k,l}/\mu_l \) for \( k = 1, \ldots, d-1 \) and \( l = 1, \ldots, n \)

In practice, as explained in the previous remark, we may dispense with step 4 and solve an \( N \times N \) generalized eigenvalue problem and associated Vandermonde system. The components of the solution of the latter will be rounded to the nearest integers and eigenvalues that have “multiplicity” zero are thrown away.
Remark. What happens if our algorithm is applied in case the $d$th components of the zeros of $f$ that lie in $D$ are not mutually distinct? In this case only a subset of the zeros will be determined correctly, namely the zeros whose $d$th component occurs only once. To illustrate what happens to the other zeros, suppose for example that $z_d^{(1)} = z_d^{(2)}$ whereas $z_d^{(3)}, \ldots, z_d^{(n)}$ are mutually distinct and
$$\{z_d^{(1)}\} \cap \{z_d^{(3)}, \ldots, z_d^{(n)}\} = \emptyset.$$ Then
$$s_p = (\nu_1 + \nu_2) [z_d^{(1)}]^p + \nu_3 [z_d^{(3)}]^p + \cdots + \nu_n [z_d^{(n)}]^p$$
for $p = 0, 1, 2, \ldots$ and
$$t_{k,p} = (\nu_1 z_k^{(1)} + \nu_2 z_k^{(2)}) [z_d^{(1)}]^p + \nu_3 z_k^{(3)} [z_d^{(3)}]^p + \cdots + \nu_n z_k^{(n)} [z_d^{(n)}]^p$$
for $k = 1, \ldots, d-1$ and $p = 0, 1, 2, \ldots$. It follows that our algorithm will replace each group of zeros that have the same $d$th component by a point in $\mathbb{C}^d$ whose $d$th component is equal to the $d$th component that is shared by these zeros and whose other components are given by the arithmetic mean of the corresponding components of these zeros.

4. Numerical examples

Let us discuss a few numerical examples.

Example 19. We have chosen $d = 4$ (the number of variables) and $n = 5$ (the number of mutually distinct zeros). The zeros and their respective multiplicities are listed in the following table.

<table>
<thead>
<tr>
<th>$z_d^{(k)}$</th>
<th>$\nu_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, i, -2, 1)</td>
<td>1</td>
</tr>
<tr>
<td>(-3, 0, 2, 2)</td>
<td>2</td>
</tr>
<tr>
<td>(2, -i, 5, 3)</td>
<td>1</td>
</tr>
<tr>
<td>(1, 0, -1, 4)</td>
<td>2</td>
</tr>
<tr>
<td>(0, 1, 3, 5)</td>
<td>1</td>
</tr>
</tbody>
</table>

Then $N = 7$ (the total number of zeros) and $z_d = (1 \cdot 1 + 2 \cdot 2 + 1 \cdot 3 + 2 \cdot 4 + 1 \cdot 5)/7 = 3$ (the arithmetic mean of the last components). The shifted moments for the last components are given by

$$\hat{s}_p = 1 \cdot (1 - 3)^p + 2 \cdot (2 - 3)^p + 1 \cdot (3 - 3)^p + 2 \cdot (4 - 3)^p + 1 \cdot (5 - 3)^p$$

and thus $\hat{s}_p = (2 + 2^p)(1 + (-1)^p)$ for $p \geq 1$. We computed the eigenvalues of the pencil

$$[\hat{s}_{l+1}]_{k,l=0}^{N-1} - \lambda [\hat{s}_{l+1}]_{k,l=0}^{N-1}$$

using LAPACK’s routine CGEGV [12]. The calculations were performed in single precision arithmetic on an IBM SP2. The results are shown in the following table, in which each generalized eigenvalue is represented as a pair $(\alpha_k, \beta_k)$ of two corresponding diagonal entries in the generalized Schur decomposition of $[\hat{s}_{l+1}]_{k,l=0}^{N-1}$ and $[\hat{s}_{l+1}]_{k,l=0}^{N-1}$.
The previous example illustrates how our algorithm is able to compute the unknown zeros and multiplicities from the integrals \( \delta_p \) and \( \hat{t}_{k,p} \). As already mentioned in Section 1, the efficient numerical evaluation of these integrals is a problem that remains to be tackled. It represents a challenge for the numerical integration community. Nevertheless we would like to present a small example in which these integrals have been calculated numerically.

**Example 20.** Consider the problem of computing all the zeros of

\[
f = f(z_1, z_2) = (\sin z_1 + z_1^2 + e^{z_2} - \cos(2z_2), \cos z_1 + z_2^3 + e^{2z_2} - 2)
\]

that lie in the polydisk

\[
D = \{ z_1 \in \mathbb{C} : |z_1| < 1 \} \times \{ z_2 \in \mathbb{C} : |z_2| < 1 \}.
\]

In this case we have to integrate over the unit cube in \( \mathbb{R}^3 \). We tried several numerical integration strategies:

- lattice rules (see for example Joe and Sloan [164, 165, 255], Beckers and Haegemans [22] and also Sidi [254] and Laurie [197]);
- Monte Carlo methods (see, e.g., Kalos and Whitlock [171] and Fishman [81]);
- the software package DCUHRE [26], written by Berntsen, Espelid and Genz. DCUHRE gave the best results. This package implements an adaptive algorithm for numerical integration over hyperrectangular regions. First we calculated \( \text{Re } I(1) \). We requested an absolute accuracy of 0.1 and obtained that \( \text{Re } I(1) \approx 1.998 \). Thus \( N = 2 \). Next we calculated the arithmetic mean of the second components. A crude approximation for this mean is sufficient to reduce ill-conditioning and therefore we requested a relative accuracy of only 0.1. Finally we calculated all the other integrals needed by our algorithm. One of the very interesting features of DCUHRE
is that it is able to integrate a vector of similar integrals over a common integration region. Since a significant part of the computation required for each integrand is the same for all of the integrands, these common calculations need to be done only once for each integrand evaluation point. We requested a relative accuracy of $10^{-5}$. DCUHRE needed about $10^5$ functional evaluations to obtain this accuracy. With these approximations for the integrals $\hat{s}_p$ and $\hat{t}_{k,p}$ as input, our algorithm obtained that $f$ has two zeros in $D$ ($n = 2$), each of multiplicity one. We refined the approximations for these zeros iteratively via Newton’s method. The zeros of $f$ that lie in $D$ are given by $(0, 0)$ and $(-0.72011062161456, 0.11033979708375)$. 

**Acknowledgements**

I had interesting conversations with Frank Dillen about differential forms and stimulating discussions with Ronald Cools and Alan Genz on the logarithmic residue integrals that appear in this chapter. I would like to thank Johan Quaegbeur for reading the proof of Theorem 37.
Part 2
CHAPTER 5

Computing simple zeros of Bessel functions

In this chapter, we consider the problem of computing simple zeros of analytic functions. In particular, we focus on the Bessel functions $J_\nu(z), Y_\nu(z), H^{(1)}_\nu(z)$ and $H^{(2)}_\nu(z)$, or their first derivative, in case the argument $z \in \mathbb{C} \setminus (-\infty, 0]$ and the order $\nu \in \mathbb{R}$. We present a software package, called ZEBEC\(^1\), for computing all the zeros of one of these functions that lie inside a rectangle whose edges are parallel to the coordinate axes. In Chapter 6 we will consider the functions $J_{n}(z) \pm iJ_{n+1}(z)$ and $[J_{n+1}(z)]^2 - J_{n}(z)J_{n+2}(z)$ in case $n \in \mathbb{N}$. The zeros of these functions play an important role in certain physical applications. We will prove that all the zeros that lie in $\mathbb{C}_0$ are simple. As ZEBEC can be easily extended to calculate zeros of any analytic function, provided that the zeros are known to be simple, it is the package of choice to calculate the zeros of $J_{n}(z) \pm iJ_{n+1}(z)$ or $[J_{n+1}(z)]^2 - J_{n}(z)J_{n+2}(z)$. Numerical examples will be given.

This chapter corresponds to our paper [185].

1. Introduction

The Bessel equation

$$z^2 u''(z) + zu'(z) + (z^2 - \nu^2)u(z) = 0, \quad z, \nu \in \mathbb{C},$$  \hspace{1cm} (52)

appears in many problems of mathematical physics. Two linearly independent solutions are given by the Bessel function of the first kind of order $\nu$

$$J_{\nu}(z) := \sum_{k=0}^{\infty} \frac{(-1)^k}{k! \Gamma(k + \nu + 1)} \left( \frac{z}{2} \right)^{\nu + 2k}, \quad z \in \mathbb{C} \setminus (-\infty, 0],$$

and the Bessel function of the second kind of order $\nu$ (also called Neumann’s function)

$$Y_{\nu}(z) := \frac{J_{\nu}(z) \cos(\nu \pi) - J_{-\nu}(z)}{\sin(\nu \pi)}, \quad z \in \mathbb{C} \setminus (-\infty, 0], \quad \nu \in \mathbb{C} \setminus \mathbb{Z}.$$  \hspace{1cm} (53)

For integer $\nu$ the right-hand side becomes indeterminate, and in this case

$$Y_{n}(z) := \lim_{\nu \to n} Y_{\nu}(z), \quad n \in \mathbb{Z}.$$  

Also of interest are the Bessel functions of the third kind, or Hankel functions,

$$H^{(1)}_{\nu}(z) := J_{\nu}(z) + iY_{\nu}(z) \quad \text{and} \quad H^{(2)}_{\nu}(z) := J_{\nu}(z) - iY_{\nu}(z).$$

\(^1\)Our package is part of the CPC Program Library. Information on how to obtain ZEBEC can be found at url http://www.cpc.cs.qub.ac.uk/cpc/cgi-bin/linet_summary.pl?CatNumber=AD10.
The above considered functions are analytic with respect to $z$ in the complex plane cut along the non-positive real axis and entire functions of the order $\nu$ for fixed $z$. They also have several other interesting features [295].

The zeros and turning points of Bessel functions are important in many branches of physical sciences and technology. They appear in the problem of cyclic membrane vibrations, the temperature distribution in a solid cylinder or in a solid sphere, the diffraction of a plane electromagnetic wave by a conducting cylinder, quantum billiards, etc. For an electrostatic interpretation of the zeros of Bessel functions, we refer to [220]. See also [219].

Some theoretical results about these zeros and turning points are available: inequalities, bounds, regions of existence or non-existence, power series expansions, Chebyshev series expansions, and qualitative results concerning their location [2, 24, 56, 151, 152, 153, 179, 180, 198, 236, 237, 238, 239, 295].

Hurwitz’s Theorem, for example, gives information about the zeros of $J_\nu(z)$ in case the order $\nu$ is real.

**Theorem 43.** Let $\nu$ be an arbitrary real number. Then the function $J_\nu(z)$ has an infinite number of positive real zeros, and a finite number $2N(\nu)$ of conjugate complex zeros, where:

- $N(\nu) = 0$ if $\nu > -1$ or $\nu = -1, -2, \ldots$,
- $N(\nu) = \nu - m + 1$ if $-m(m + 1) < \nu < -m, m = 1, 2, \ldots$.

In the second case, if $[-\nu]$ is odd, then there is a pair of purely imaginary zeros among the conjugate complex zeros.

In [240, 242] a software package was presented for computing zeros of $J_\nu(x)$, in case $x > 0$ and $\nu > -1$, and turning points of $J_\nu(x)$, i.e., zeros of $J'_\nu(x)$, in case $x > 0$ and $\nu > 0$. The package RFSFNS [291] can be used to calculate zeros and turning points of $J_\nu(x)$ and $Y_\nu(x)$, in case $x > 0$ and $\nu \geq 0$. Recently, Segura and Gil [253] have presented two programs to calculate real zeros of $J_\nu(x)$ for real orders $\nu$ (positive or negative). Their algorithm proceeds by applying Newton’s method to the monotonic function $x^{2\nu-1}J_\nu(x)/J_{\nu-1}(x)$. To evaluate this function, they use the continued fraction representation of the ratio $J_\nu(x)/J_{\nu-1}(x)$ and apply the Lentz-Thompson algorithm. Ikhe et al. [154, 155] showed how the complex zeros of $J_\nu(z)$ can be calculated from an eigenvalue problem for infinite compact complex symmetric matrices. In this chapter we present a reliable and portable package for computing zeros or turning points of $J_\nu(z), Y_\nu(z), H^{(1)}_\nu(z)$ and $H^{(2)}_\nu(z)$, in case the argument $z$ belongs to the complex plane cut along the non-positive real axis and the order $\nu$ is real. The package is called ZEBEC: Zeros of Bessel functions Complex. ZEBEC is capable of calculating all the zeros or turning points that lie inside a rectangle whose edges are parallel to the coordinate axes. It is known that any solution of (52) has only simple zeros, except possibly at $z = 0$ (see, e.g., [118, p. 79] or [295, p. 479]). In particular, this holds for $J_\nu(z), Y_\nu(z), H^{(1)}_\nu(z)$ and $H^{(2)}_\nu(z)$. Besides, the derivative of any solution of (52) also has only simple zeros, except possibly at $z = 0$ or $z = \pm \nu$ [175, 176, 267]. Therefore, the zeros that we set out to compute are simple, except possibly at $z = 0$ and $z = \pm \nu$. As we restrict the argument $z$ to $\mathbb{C} \setminus (-\infty, 0]$ and the order $\nu$ to $\mathbb{R}$, we do not have to
consider the point $z = 0$ and one of the points $z = \pm \nu$. By evaluating the function and its first derivative at $z = \nu$ (in case $\nu > 0$) or $z = -\nu$ (in case $\nu < 0$), one can easily verify whether the function has a multiple zero at this point. If this is indeed the case, then ZEBEC notifies the user and asks for a different rectangular region. Else, the computation starts.

2. Computing simple zeros of analytic functions

Given a rectangle whose edges are parallel to the coordinate axes, we take the following approach.

(1) We calculate the total number of zeros that lie inside this rectangle.

(2) We isolate all these zeros via consecutive subdivisions. In other words, we obtain a set of subrectangles, each of which contains precisely one zero.

(3) For each of these subrectangles, we calculate the zero that lies inside it.

For the first two phases we use logarithmic residue integrals, whereas for the third phase we use a generalized method of bisection. All these ingredients will be briefly explained below.

Let $W$ be a simply connected region in $\mathbb{C} \setminus (-\infty, 0]$, $f$ one of the Bessel functions mentioned in Section 1, and $\gamma$ a positively oriented Jordan curve in $W$ that does not pass through any zero of $f$. As $f$ is analytic in $W$, the total number of zeros of $f$ that lie inside $\gamma$ is given by

$$ N := \frac{1}{2\pi i} \int_{\gamma} \frac{f'(z)}{f(z)} dz. $$

Suppose that a parametric representation of the curve $\gamma$ is given by $z = \gamma(t)$, $a \leq t \leq b$, and that the mapping $\gamma : [a, b] \to \mathbb{C}$ is piecewise continuously differentiable in $[a, b]$. Thus, a partition $a = t_0 < t_1 < \cdots < t_{r-1} < t_r = b$ of the interval $[a, b]$ can be found, such that the restriction of $\gamma$ to each subinterval $[t_{j-1}, t_j]$ is continuously differentiable.

Let $W^s := \{(x, y) \in \mathbb{R}^2 : x + iy \in W\}$ and $u, v : W^s \to \mathbb{R}$, with

$$ u(x, y) := \text{Re} f(x + iy) \quad \text{and} \quad v(x, y) := \text{Im} f(x + iy). $$

By restricting $u$ and $v$ to the curve $\gamma$, we obtain

$$ u(t) := u(x(t), y(t)) \quad \text{and} \quad v(t) := v(x(t), y(t)), $$

where $x(t) := \text{Re} \gamma(t)$, $y(t) := \text{Im} \gamma(t)$ and $t \in [a, b]$. Then, by using the Cauchy-Riemann equations for $u$ and $v$, one can easily verify that

$$ N = \frac{1}{2\pi} \sum_{j=1}^{r} \int_{t_{j-1}}^{t_j} \frac{u \frac{dv}{dt} - v \frac{du}{dt}}{u^2 + v^2} dt, $$

where we have considered the above partition of $[a, b]$. This formula for $N$ is the Kronecker integral for the topological degree of the real mapping $F = (u, v)$ at the origin relative to the interior of $\gamma$. For an introduction to degree theory we refer the interested reader to Lloyd [201]. Also, for a detailed description of the Kronecker integral, see Hoenders and Slump [144, 145].
Let us consider the case that $\gamma$ is a rectangle whose edges are parallel to the coordinate axes. Suppose that it has left lower vertex $(x_0, y_0)$ and that its edges have length $h_1$ and $h_2$. In other words, suppose that $\gamma$ is the boundary of $[x_0, x_0 + h_1] \times [y_0, y_0 + h_2]$.

Define the functions $\psi$ and $\varphi$ in $W^*$ as

$$\psi(x, y) := \frac{u(x, y)v_y(x, y) - v(x, y)u_y(x, y)}{[u(x, y)]^2 + [v(x, y)]^2}$$

and

$$\varphi(x, y) := \frac{u(x, y)v_y(x, y) - v(x, y)u_y(x, y)}{[u(x, y)]^2 + [v(x, y)]^2}.$$

Then one can easily verify that

$$N = h_1 (I_1 - I_3) + h_2 (I_2 - I_4)$$

with

$$I_1 := \frac{1}{2\pi} \int_0^1 \psi(x_0 + th_1, y_0) \, dt$$

$$I_2 := \frac{1}{2\pi} \int_0^1 \varphi(x_0 + h_1, y_0 + th_2) \, dt$$

$$I_3 := \frac{1}{2\pi} \int_0^1 \psi(x_0 + th_1, y_0 + h_2) \, dt$$

$$I_4 := \frac{1}{2\pi} \int_0^1 \varphi(x_0, y_0 + th_2) \, dt.$$

These formulae are the basis of the phases (1) and (2) of our method.

As $f$ may have zeros on the boundary of the rectangular box specified by the user, the algorithm starts by perturbing this box. For this purpose a tolerance is used that is taken to be proportional to a power of the machine precision, for example $10$ times the square root of the machine precision. The box is then slightly enlarged in an asymmetrical way. The reason for this asymmetric perturbation is to lower the possibility of having a zero close to or on any boundary of the consecutive subdivisions. For example, if the starting box is symmetric with respect to the imaginary axis, then the inner boundary at the first subdivision will pass through any imaginary zeros of $f$.

The total number of zeros of $f$ that lie inside the perturbed box is obtained by calculating the integrals $I_1, \ldots, I_4$ via the adaptive integrator DQAG from the package QUADPACK [241]. A zero near one of the edges of the rectangle causes the integrand of the corresponding integral to have a "peak." The closer the zero lies to the edge, the sharper this peak is. If the zero lies on the edge, then the integral is divergent. DQAG uses adaptive strategies that enable it to cope with such peaks efficiently. However, if a zero lies too close to an edge (the corresponding peak is too sharp), then DQAG warns us that it has problems in calculating the integral. Our algorithm then slightly moves this edge and restarts. By enlarging the user's
box, we may of course include additional zeros. We have decided not to discard any of these zeros ourselves. Rather, we provide the user with the box that eventually has been considered, all the zeros that lie inside this box, and leave it to him/her to filter out unwanted zeros.

If the starting box (as perturbed by the method) contains exactly one zero, then this zero is calculated via the package CHABIS [288, 289]. This package implements a generalized method of bisection, known as characteristic bisection, which we will explain below. Otherwise, the longest edges of the box are halved and the box is subdivided into two equal boxes. The number of zeros in each of these boxes is calculated via numerical integration. If DQAG detects a zero near the inner edge, then this edge is shifted, a process that results in an asymmetric subdivision of the box. Then the two smaller boxes are examined. A box that does not contain any zero is abandoned. A box that contains precisely one zero is handed to CHABIS. A box that contains more than one zero is subdivided again. This process is repeated until all the zeros have been isolated — a set of boxes has been found, each of which contains precisely one zero — and computed.

The package CHABIS can be used to solve systems of nonlinear equations. It is based on the topological degree. Let us briefly describe how it works. First, what is known as a characteristic polyhedron (CP) is constructed. Let $F: \mathbb{R}^d \to \mathbb{R}^d$ be the mapping that defines the nonlinear system. Every polyhedron that has $2^d$ vertices and is such that the signs of the components of $F$ evaluated at these vertices produce all the possible combinations of $-1$ and $1$ is by definition a CP for $F$. Under certain assumptions on the boundary of a CP, this property implies that the topological degree of $F$ at the origin relative to the interior of the CP is different from zero. Kronecker’s Theorem [201] then implies that $F$ has at least one zero in the interior of the CP. Note that the topological degree of $F$ relative to the interior of the CP does not have to be computed explicitly. Given a CP for $F$, a smaller CP can be constructed by replacing a certain vertex of the CP by the midpoint of its longest edge. The vertex that will be replaced is such that the signs of the components of $F$ at this vertex coincide with those at the midpoint of the longest edge. The new polyhedron is again a CP, and thus it also contains at least one zero of $F$. This refinement process, which is called characteristic bisection, is repeated until the diameter of the polyhedron is smaller than a given tolerance.

The approach taken by CHABIS has several advantages. First, as soon as an initial characteristic polyhedron has been found, the convergence is guaranteed. The zeros of $F$ have to be regular though. (A zero $x^* \in \mathbb{R}^d$ of $F$ is called regular if the Jacobian matrix of $F$ at $x^*$ is nonsingular.) Otherwise, the theory of the topological degree is not applicable. A second advantage is that the number of iterations needed to approximate a zero to a certain accuracy is known a priori. Furthermore, as the method depends only on the signs of the components of $F$, it is suitable for solving problems with imprecise function values or involving infinite series expansions. For example, in the case of Bessel and Airy functions Vrahatis et al. [290, 293] showed that the sign stabilizes after summing a relatively small number of terms of the power series, which leads to a considerable speed-up of the calculations. A theoretical approach to the problem of computing complex zeros of Bessel functions by using only the signs of the respective power series was presented in [292].
As already mentioned in Section 1, the functions $J_\nu(z)$, $Y_\nu(z)$, $H^{(1)}_\nu(z)$ and $H^{(2)}_\nu(z)$ may have a multiple zero at $z = \pm \nu$. As our algorithm can manage only simple zeros, we have to examine if such a multiple zero exists inside the rectangular region specified by the user. This can be done very easily: if the rectangular box contains $z = \nu$ (in case $\nu > 0$) or $z = -\nu$ (in case $\nu < 0$), then the algorithm evaluates the function and its first derivative at this point. If these values are both very small, then ZEBEC warns the user and asks for a different rectangular region. Else, the computation starts.

3. The packages BESSCC, QUADPACK and CHABIS

In this section we give a brief outline of the packages that ZEBEC uses: BESSCC, QUADPACK and CHABIS.

BESSCC. This package was written by Thompson and Barnett [268]. It computes the modified Bessel functions $I_\nu(z)$ and $K_\nu(z)$ as well as their first derivative for complex argument $z$ and real order $\nu$. To obtain $J_\nu(z)$ from these functions, we have used the following analytic continuation and reflection formulae [2]: if $\nu \geq 0$, then

$$J_\nu(z) = \begin{cases} e^{i\pi \nu} I_\nu(-iz) & \text{if } \Im z \geq 0, \\ e^{-i\pi \nu} I_\nu(iz) & \text{if } \Im z < 0, \end{cases}$$

else

$$J_\nu(z) = J_{-\nu}(z) \cos(\pi \nu) + Y_{-\nu}(z) \sin(\pi \nu).$$

For $Y_\nu(z)$ the following holds: if $\nu \geq 0$, then

$$Y_\nu(z) = \begin{cases} i e^{i\pi \nu} I_\nu(-iz) - \frac{\pi}{i} e^{-i\pi \nu} K_\nu(-iz) & \text{if } \Im z \geq 0, \\ \overline{Y_\nu(z)} & \text{if } \Im z < 0, \end{cases}$$

else

$$Y_\nu(z) = Y_{-\nu}(z) \cos(\pi \nu) - J_{-\nu}(z) \sin(\pi \nu).$$

We have evaluated the Bessel functions of the third kind by using their definition (53). To obtain formulae for the derivatives, we have differentiated the previous identities.

For compatibility purposes, and with the permission of the authors, we have also made a few small changes to BESSCC.

QUADPACK. This package was written by Piessens et al. [241]. It is a widely used package for automatic integration. It consists of 12 quadrature routines. Here the routine DQAG is used, which implements a globally adaptive integrator for calculating definite integrals over finite intervals. DQAG is based on Gauss-Kronrod quadrature rules. QUADPACK is written in ANSI standard Fortran 77.

CHABIS. This package was written by Vrahatis [288]. It can be used to solve nonlinear systems of equations in case the only computable information consists of the algebraic signs of the components of the mapping that defines the nonlinear system. CHABIS starts by locating at least one solution that lies inside a polyhedron. Then it applies a generalized method of bisection to this polyhedron. The package
consists of nine subprograms, one of which is called by the user. CHABIS is written in ANSI standard Fortran 77.

4. The package ZEBEC

The package ZEBEC (‘Zeros of BEssel functions Complex’) contains about 7000 lines of code including comments. It is written in Fortran 77 and has been thoroughly tested on various UNIX machines (DEC DS5000/7012, HP 9000 B160L, IBM RS6000 7012, SUN SPARC Ultra-2 m1170) as well as on a PC IBM compatible.

ZEBEC consists of seven parts, namely, the main program ZEBEC, the subroutines MANAGE, INBOX, SPLIT, RCOMP, and FDF and the function FNC, and a set of functions related to the integrals \( I_1, \ldots, I_4 \). ZEBEC also requires the subroutine DQAG from the package QUADPACK [241], the package CHABIS [288] and the package BESSCC [268] from the CPC Program Library.

In the main program ZEBEC the following parameters have to be set:

**MAXRT**: a positive integer that determines the maximum number of zeros that may be requested.

**ICASE**: an integer in \( \{1, \ldots, 8\} \) that specifies which function is to be considered:
- 1. \( J \), the Bessel function of the first kind,
- 2. the derivative of \( J \),
- 3. \( Y \), the Bessel function of the second kind,
- 4. the derivative of \( Y \),
- 5. \( H^{(1)} = J + iY \), the Bessel function of the third kind,
- 6. the derivative of \( H^{(1)} \),
- 7. \( H^{(2)} = J - iY \), the Bessel function of the third kind,
- 8. the derivative of \( H^{(2)} \).

**XNU**: a real variable that specifies the order of the Bessel function.

**XO**: a real array of length 2 that contains the \( x \)- and \( y \)-coordinates of the left lower vertex of the rectangle that is to be examined.

**H**: a real array of length 2 that specifies the size of this rectangle along the \( x \)- and \( y \)-direction.

**ICON**: an integer in \( \{1, \ldots, 4\} \) that specifies which calculations are to be done:
- 1. calculation of the total number of zeros, only,
- 2. calculation of the total number of zeros and isolation of each one of them,
- 3. calculation of the total number of zeros, isolation and computation of each one of them,
- 4. calculation of the total number of zeros; isolation and computation of \( NR \) zeros.

Note that if \( ICON=4 \), then the user also has to supply the value of the requested number of zeros \( NR \).
EPSIL0: a real variable that is used in the stopping criterion for the computation of the zeros. Termination occurs if the algorithm estimates that the infinity norm of the mapping $F := (u, v)$, where $u := \Re f$ and $v := \Im f$ and $f$ is one of the Bessel functions or their first derivative, at an approximate zero is at most EPSIL0 or if the size of the box that contains an approximate zero is at most $4\cdot 10^{-6}\cdot$EPSIL0.

If EPSIL0 is set to a value that is less than the machine precision EPSMCH, then EPSIL0 is set equal to $5\cdot 10^{-6}\cdot$EPSMCH. EPSMCH is computed within ZEBEC.

EPSABS: a real variable that determines the absolute accuracy to which the integrals are to be evaluated. If EPSABS=0.0, then only a relative precision criterion will be used.

EPSREL: a real variable that determines the relative accuracy to which the integrals are to be evaluated. If EPSREL=0.0, then only an absolute precision criterion will be used.

If EPSABS and EPSREL are both too small, then the numerical integration may be time consuming. If they are both too large, then the calculated number of zeros may be wrong. The default values of EPSABS and EPSREL are 0.07 and 0.00, respectively.

ACC: a variable whose value determines a target relative accuracy for BESSCC. If ACC is larger than 0.001 or less than the machine precision, then it is set equal to its default value ACCDEF=10E-6.

Let us briefly describe the various parts of ZEBEC.

The subroutine INBOX calculates the total number of zeros that lie inside the rectangular box given by the user. If some of the zeros lie too close to the boundary of this box and the quadrature routine DQAG fails, then INBOX perturbs the box slightly and enlarges it.

The subroutine SPLIT takes a box and splits it into two boxes. A symmetric splitting, which proceeds by halving the longest edges, is tried first. If the calculation of the integral along the inner edge fails, then it is assumed that some of the zeros lie too close to this edge, and the inner edge is shifted.

The subroutine RCOMP takes a box that contains exactly one zero and returns an approximation for this zero. RCOMP calls CHABIS.

The subroutine MANAGE forms the main part of the package. MANAGE starts by calling INBOX. If there are no zeros inside the user’s box, then the program stops. If there is exactly one zero inside this box, then RCOMP is called (if ICON=3 or 4). Else, the box is given to SPLIT. The two boxes returned by SPLIT are examined. A box that does not contain any zero is abandoned. A box that contains exactly one zero is given to RCOMP (if ICON=3 or 4). A box that contains more than one zero is put in a list. Then MANAGE takes the next box from this list and calls SPLIT. This procedure is repeated until all the requested zeros are isolated (if ICON=2, 3 or 4) and computed (if ICON=3 or 4).

DQAG requires function and derivative values. These are provided by the subroutine FDF. CHABIS needs only function values. These are provided by the function FNC. Both FDF and FNC call BESSCC.
The program execution terminates normally after the completion of its task. This type of termination is indicated by the value 1 of the output variable INFO. If the value of this parameter is different from 1, then the termination of the program is abnormal. The cases of abnormal termination are the following:

INFO=0: Improper input parameters:
- the input values of ICASE or ICON are out of range, or
- NR exceeds MAXRT, or
- the box specified by X0 and H crosses the non-positive real axis, or
- the box specified by the user contains a multiple zero at \( z = \nu \) or \( z = -\nu \) (ICASE=2, 4, 6 or 8), or
- H(1) or H(2) is negative, or
- EPSABS or EPSREL is negative.

INFO=2: The procedure for the calculation of the total number of zeros has failed.

INFO=3: The procedure for the isolation of the zeros has failed.

INFO=4: The procedure for the computation of the zeros has failed.

Upon normal termination, the main output values of the program are given by the following parameters of MANAGE:

NRPERT: an integer that is equal to the number of zeros that exist in the examined box in case ICON is not equal to 3.

LFLRT: an integer that is equal to the number of zeros that exist in the examined box in case ICON is equal to 3.

XOFIN: a real array of size 2xMAXRT that contains the \( x \)- and \( y \)-coordinates of the left lower vertices of the rectangles that have been found to contain exactly one zero.

HFIN: a real array of size 2xMAXRT that specifies the size of these rectangles along the \( x \)- and \( y \)-direction.

ROOTS: a real array of size 2xMAXRT that contains the real and imaginary parts of the zeros that have been found in the examined box.

FROOTS: a real array of size 2xMAXRT that contains the values of the function (real and imaginary parts) at the approximations for the zeros that have been found.

5. An example of how to use ZEBEC

Let us demonstrate how ZEBEC can be used to calculate the total number of zeros that lie inside a rectangular box, to isolate these zeros, and to compute them. Suppose that we want to calculate five zeros of \( Y_{-15.3}(z) \) that lie inside the box

\[
\{ z \in \mathbb{C} : -22 \leq \text{Re} z \leq 23, \quad 0.5 \leq \text{Im} z \leq 100.5 \}.
\]

The corresponding input values are: ICASE=3, XNU=-15.3D0, ICON=4, NR=5, and X0(1)=-22.D0, X0(2)=0.5D0, H(1)=45.D0, H(2)=100.D0. We request an accuracy of EPSIL=1.0D-13.

For this example the main program of ZEBEC is the following:
PROGRAM ZEBEC
IMPLICIT NONE

INTEGER MAXRT
PARAMETER (MAXRT = 100)

INTEGER IFLRT(MAXRT), ICASE, ICON, NR, INFO, NRKEEP, IBESS
INTEGER J, LFLRT, NRPERT, IMAXRT

DOUBLE PRECISION XO(2), H(2), XOPERT(2), HPERT(2)
DOUBLE PRECISION ROOTS(2,MAXRT), FRROOTS(2,MAXRT)
DOUBLE PRECISION POINTS(2,MAXRT), STEPS(2,MAXRT), NRS(MAXRT)
DOUBLE PRECISION XOFIN(2,MAXRT), HFIN(2,MAXRT)
DOUBLE PRECISION RINTS(6,MAXRT), ERRS(6,MAXRT)
DOUBLE PRECISION EPSILO, XNU, EPSABS, EPSREL, ACC

EXTERNAL MANAGE

COMMON /BLK1/ ICON, NR
COMMON /BLK2/ INFO
COMMON /BLK3/ XNU
COMMON /BLK4/ ICASE
COMMON /BLK5/ IBESS
COMMON /BLK6/ EPSABS, EPSREL
COMMON /BLK7/ ACC
COMMON /BLK8/ IMAXRT

DATA ICASE, XNU, ICON, NR
+ / 3, -15.3D0, 4, 5 /
DATA XO(1), XO(2), H(1), H(2)
+ / -22.0D0, 0.5D0, 45.0D0, 100.0D0 /

EPSILO = 1.0D-13
EPSABS = 0.07D0
EPSREL = 0.0D0
ACC = 1.0D-13

PRINT 9999, ICASE, XNU, XO, H, ICON, EPSILO
NRKEEP = NR
IBESS = 0
IMAXRT = 0
CALL MANAGE(MAXRT,XO,H,XPERT,EPERT,EPERT,EPERT,EPSLO,EPERT,
  \hspace{1cm} \hspace{1cm} + XOFIN,HFIN,ROOTS,ROOTS,POINTS,POINTS,POINTS,
  \hspace{1cm} \hspace{1cm} + RINTS,ERRS,NRS,IFLRT,IFLRT)

IF ( INFO .EQ. 0 ) THEN
  PRINT 9998
  GO TO 10
END IF

IF ( IBESS .EQ. 2 ) PRINT 9997

IF ( INFO .EQ. 2 ) THEN
  PRINT 9996
  GO TO 10
END IF

IF ( IMAKT .EQ. 1 ) THEN
  PRINT 9995, MAXRT
  STOP
END IF

PRINT 9994, XOPERT(1), XOPERT(2), HPERT(1), HPERT(2)

IF ( ICON .NE. 3 ) THEN
  PRINT 9993, NRPERT
ELSE
  PRINT 9993, LFLRT
END IF

IF ( NRPERT .EQ. 0 ) GO TO 10
IF ( ICON .EQ. 1 ) GO TO 10

IF ( INFO .EQ. 3 ) THEN
  PRINT 9992
  GO TO 10
ENDIF

IF ( ICON .EQ. 4 ) THEN
  PRINT 9991, NRKEEP, NR
ELSE
  PRINT 9990, LFLRT
END IF

PRINT 9989
DO J = 1,LFLRT
  PRINT 9988, J, XOFIN(1,J), XOFIN(2,J), HFIN(1,J), HFIN(2,J)
COMPUTING SIMPLE ZEROS OF BESSEL FUNCTIONS

END

IF ( ICON .EQ. 2 ) GO TO 10

IF ( INFO .EQ. 4 ) THEN
  PRINT 9987
  GO TO 10
ENDIF

PRINT 9986
DO J = 1, LFLRT
  IF ( IFLRT(J) .EQ. 1 ) THEN
    PRINT 9985, J, ROOTS(1,J), ROOTS(2,J),
    + FROOTS(1,J), FROOTS(2,J)
  ELSE
    PRINT 9984, J
  END IF
END DO
END IF

10 PRINT 9983, INFO
STOP

9999 FORMAT (/2X, 'STARTING VALUES :', /3X, 17('"'),
+ /2X, 'ICASE : ', I1,
+ /2X, 'ORDER : ', F22.15,
+ /2X, 'XO : ', F22.15, F23.15,
+ /2X, 'H : ', F22.15, F23.15,
+ /2X, 'ICON : ', I1,
+ /2X, 'EPSILO : ', F22.15,
+ /3X, 67('"'),
+ /2X, 'RESULTS :', /3X, 9('"'))

9998 FORMAT (/2X, '*** IMPROPER INPUT PARAMETERS ***')

9997 FORMAT (/2X, 'THE PROCEDURE FOR THE CALCULATION OF THE',
+ 'BESSEL',
+ /2X, 'FUNCTION FAILED. THE RESULTS MAY BE',
+ 'INACCURATE **')

9996 FORMAT (/2X, '*** THE PROCEDURE FOR THE CALCULATION OF',
+ /2X, 'THE TOTAL NUMBER OF ZEROS FAILED ***')

9995 FORMAT (/2X, '*** THE NUMBER OF ZEROS EXCEEDS MAXRT = ', I5,
+ /2X, 'INCREASE THE VALUE OF MAXRT ***')

9994 FORMAT (/2X, 'THE FOLLOWING BOX WAS CONSIDERED:')
ZEBEC outputs the following:

STARTING VALUES:
------------------

ICASE : 3
ORDER : -15.30000000000001

X0 : -22.00000000000002, .50000000000000
H : 45.00000000000000, 100.00000000000000
ICON : 4
EPSILO : .00000000000100

RESULTS :
--------

THE FOLLOWING BOX WAS CONSIDERED:

\[
X_0 = \begin{pmatrix} 22.00000016391274 \\ 45.000000357627865 \end{pmatrix}, \quad \epsilon = \begin{pmatrix} 84904 \\ 17232520 \end{pmatrix}
\]

THE TOTAL NUMBER OF ZEROS WITHIN THIS BOX IS : 16

NUMBER OF ZEROS REQUESTED : 5

NUMBER OF ZEROS ISOLATED : 5

BOXXES CONTAINING A SINGLE ZERO :
-----------------------

1) \[
X_0 = \begin{pmatrix} 7.500000104308129 \\ 250000089406966 \end{pmatrix}, \quad \epsilon = \begin{pmatrix} 999806284904 \\ 50000052154063 \end{pmatrix}
\]

2) \[
X_0 = \begin{pmatrix} 0.00000014901161 \\ 250000089406966 \end{pmatrix}, \quad \epsilon = \begin{pmatrix} 999806284904 \\ 50000026077032 \end{pmatrix}
\]

3) \[
X_0 = \begin{pmatrix} 6.125000059604645 \\ 250000022351742 \end{pmatrix}, \quad \epsilon = \begin{pmatrix} 74999832361937 \\ 56250006519258 \end{pmatrix}
\]

4) \[
X_0 = \begin{pmatrix} 6.125000059604645 \\ 250000022351742 \end{pmatrix}, \quad \epsilon = \begin{pmatrix} 31249983881195 \\ 56250006519258 \end{pmatrix}
\]

5) \[
X_0 = \begin{pmatrix} 5.00000014901161 \\ 250000044703484 \end{pmatrix}, \quad \epsilon = \begin{pmatrix} 87499984500453 \\ 125000013038516 \end{pmatrix}
\]

FINAL APPROXIMATE ZEROS AND VERIFICATION :
-----------------------

1) \[
Z = (12.507257919321071, \quad 4.095557539693683) \\
F(Z) = (-0.00000000000003, \quad -0.00000000000039)
\]

2) \[
Z = (10.378711252301940, \quad 6.178243183678395) \\
F(Z) = (-0.00000000000009, \quad 0.00000000000031)
\]

3) \[
Z = (8.447945724224951, \quad 7.61385057712050)
\]
More numerical examples can be found in [185].

6. Concluding remarks

The package ZEBEC has been applied to Bessel functions of various orders and random boxes. We have found that it behaves predictably and accurately. It calculates with certainty the total number of zeros that lie inside a given box, isolates each one of them, and then computes all these zeros or a requested number of them.

The user will appreciate the flexibility offered by the input parameter ICON. If nothing is known about the zeros that lie inside the given box, one may call ZEBEC with ICON = 1 to obtain the total number of zeros. Then one may proceed with ICON = 3 to isolate and compute all these zeros, or, if less than the total number of zeros are required, with ICON = 4 and NR equal to the requested number of zeros. If only a set of boxes is required, each of which contains exactly one zero, then one may set ICON = 2.

Our package can be applied to any analytic function that has only simple zeros, provided that a Fortran 77 routine exists to evaluate this function and its first derivative. In fact, in Chapter 6 we will consider the functions $J_n(z) \pm iJ_{n+1}(z)$ and $|J_{n+1}(z)|^2 - J_n(z)J_{n+2}(z)$ in case $n \in \mathbb{N}$. The zeros of these functions play an important role in certain physical applications. We will prove that all the zeros that lie in $\mathbb{C}_0$ are simple. Thus ZEBEC is the package of choice to calculate approximations for these zeros.

Acknowledgements

The larger part of ZEBEC was written during two research stays (June 2–27, 1997 and March 22–April 18, 1998) at the Department of Mathematics, University of Patras (Patras, Greece). I would like to thank Michael Vrahatis, Omiros Ragos and Filareti Zafiropoulos for their very kind hospitality and for giving me the opportunity to collaborate with them. I would also like to thank Vlaamse Leergangen Leuven for its generous financial support.
On the zeros of $J_n(z) \pm i J_{n+1}(z)$ and

$$[J_{n+1}(z)]^2 - J_n(z) J_{n+2}(z)$$

The zeros of $J_n(z) \pm i J_{n+1}(z)$ and $[J_{n+1}(z)]^2 - J_n(z) J_{n+2}(z)$, where $n \in \mathbb{N}$, play an important role in certain physical applications. At the origin these functions have a zero of multiplicity $n$ (if $n \geq 1$) and $2n + 2$, respectively. We prove that all the zeros that lie in $\mathbb{C}_0$ are simple. As ZEBEC, the software package for computing zeros or turning points of Bessel functions that we presented in the previous chapter, can be easily extended to calculate zeros of any analytic function, provided that its zeros are known to be simple, it is the package of choice to calculate the zeros of $J_n(z) \pm i J_{n+1}(z)$ or $[J_{n+1}(z)]^2 - J_n(z) J_{n+2}(z)$. We tabulate the first 30 zeros of $J_5(z) - i J_6(z)$ that lie in the fourth quadrant as computed by ZEBEC.

This chapter corresponds to our paper [193].

1. Introduction

In this chapter we focus on the complex zeros of the entire functions

$$z \mapsto J_n(z) \pm i J_{n+1}(z)$$

and

$$z \mapsto [J_{n+1}(z)]^2 - J_n(z) J_{n+2}(z)$$

where $n \in \mathbb{N}$ and $J_n(z)$ denotes the Bessel function of the first kind of order $n$,

$$J_n(z) = \left(\frac{z}{2}\right)^n \sum_{k=0}^{\infty} \frac{(-1)^k}{k! (n+k)!} \left(\frac{z}{2}\right)^{2k}, \quad |z| < \infty.$$  

Let $F_n(z) := J_n(z) - i J_{n+1}(z)$ and $G_n(z) := [J_{n+1}(z)]^2 - J_n(z) J_{n+2}(z)$.

The zeros of these functions play an important role in physical applications. The zeros of $F_0(z)$ are of interest to the specialist in the problem of water wave run up on a sloping beach, cf. Synolakis [263, 264]. The equation $F_n(z) = 0$ arises in problems of wave reflection from composite beaches, i.e., beaches with multiple slopes [266]. MacDonald [205] used the zeros of $G_0(z)$ to plot representative streamlines for the steady motion of a viscous fluid in a long tube, of constant radius, which rotates about its axis (the $\hat{z}$ axis) with an angular velocity that changes discontinuously at $\hat{z} = 0$ from one constant value to another of the same sign.

MacDonald derived asymptotic formulae for the zeros of $F_0(z)$, $F_n(z)$ and $G_n(z)$ in [204], [205] and [206], resp. See also Rawlins [244]. These formulae permit to locate zeros of large modulus. MacDonald observed that the accuracy of the asymptotic formulae for the zeros of $F_n(z)$ deteriorates as $n$ increases. Also, asymptotic
formulæ may be inadequate for smaller zeros. These have to be computed by some numerical procedure. To obtain the smaller zeros of \( F_n(z) \), MacDonald \cite{206} truncated the ascending series of \( F_n(z) \) and calculated the zeros of the truncated series via Newton’s method.

We suggest to use the package ZEBC, which we presented in the previous chapter, to compute zeros of (54) or (55). Given a rectangular region in the complex plane, ZEBC is able to compute all the zeros of the Bessel functions \( J_\nu(z) \), \( Y_\nu(z) \), \( H_\nu^{(1)}(z) \), \( H_\nu^{(2)}(z) \) or their first derivatives, where \( z \in \mathbb{C} \setminus (-\infty,0] \) and \( \nu \in \mathbb{R} \), that lie inside this region. The package can be easily extended to calculate zeros of any analytic function, provided that the zeros are known to be simple. We will show that all the zeros of (54) and (55) that lie in \( \mathbb{C}_0 \) are simple. Hence ZEBC is the package of choice to calculate these zeros.

This chapter is organized as follows. In Section 2 and 3 we summarize relevant properties of the zeros of (54) and (55), respectively. At \( z = 0 \) these functions have a zero of multiplicity \( n \) (if \( n \geq 1 \)) and \( 2n + 2 \), respectively. We prove that all the zeros that lie in \( \mathbb{C}_0 \) are simple. In Section 4 we examine how ZEBC can be used to calculate zeros of (54) or (55). We tabulate the first 30 zeros of \( J_0(z) - iJ_0(z) \) that lie in the fourth quadrant as computed by ZEBC.

2. The zeros of \( J_n(z) \pm iJ_{n+1}(z) \)

If \( n \geq 1 \), then \( J_n(z) \) has a zero of multiplicity \( n \) at the origin, whereas \( J_{n+1}(z) \) has a zero of multiplicity \( n + 1 \) at the origin. Therefore, if \( n \geq 1 \), then the functions \( J_n(z) \pm iJ_{n+1}(z) \) have a zero of multiplicity \( n \) at the origin. However, they don’t have other real zeros.

**Theorem 44.** Except for \( z = 0 \), the functions \( J_n(z) \pm iJ_{n+1}(z) \) have no zeros on the real axis.

**Proof.** Suppose that \( z^* \in \mathbb{R}_0 \) is such that

\[
J_n(z^*) - iJ_{n+1}(z^*) = 0.
\]

The Taylor series (56) implies that

\[
\overline{J_n(z)} = J_n(\overline{z})
\]

and thus \( J_n(z) \) and \( J_{n+1}(z) \) take real values on the real axis. Therefore (57) cannot hold unless \( J_n(z^*) = J_{n+1}(z^*) = 0 \). However, the positive zeros of \( J_n(z) \) are interlaced with those of \( J_{n+1}(z) \), cf. Watson \cite{295, §15.22 on page 479}. Thus, no positive zero of \( J_n(z) \) coincides with a positive zero of \( J_{n+1}(z) \). Also, (56) implies that

\[
J_n(-z) = (-1)^n J_n(z)
\]

Therefore, the zeros of \( J_n(z) \) and \( J_{n+1}(z) \) are symmetric about the origin. Thus, no negative zero of \( J_n(z) \) coincides with a negative zero of \( J_{n+1}(z) \) and we may conclude that \( J_n(z^*) = J_{n+1}(z^*) = 0 \) cannot hold if \( z^* \in \mathbb{R}_0 \). This shows that \( J_n(z) - iJ_{n+1}(z) \) has no zeros in \( \mathbb{R}_0 \). The proof for \( J_n(z) + iJ_{n+1}(z) \) is analogous. □

Except for \( z = 0 \), the functions \( J_n(z) \pm iJ_{n+1}(z) \) have no zeros on the imaginary axis. This is a corollary of the following theorem.
THEOREM 45. The following integral representations hold:

\[ J_n(z) + iJ_{n+1}(z) = \frac{1}{\sqrt{\pi} \Gamma(n+\frac{3}{2})} \left( \frac{z}{2} \right)^n \int_{-1}^{1} e^{izt} (1+t)^{n+\frac{3}{2}} (1-t)^{n-\frac{1}{2}} \, dt \]

and

\[ J_n(z) - iJ_{n+1}(z) = \frac{1}{\sqrt{\pi} \Gamma(n+\frac{3}{2})} \left( \frac{z}{2} \right)^n \int_{-1}^{1} e^{izt} (1+t)^{n-\frac{1}{2}} (1-t)^{n+\frac{3}{2}} \, dt. \]

PROOF. The Poisson integral representation for Bessel functions of the first kind (see for example [77, formula (7) on page 81]) tells us that

\[ \Gamma(n+\frac{1}{2}) J_n(z) = \frac{1}{\sqrt{\pi}} \left( \frac{z}{2} \right)^n \int_{-1}^{1} e^{izt} (1-t^2)^{n-\frac{1}{2}} \, dt. \]

By replacing \( n \) by \( n+1 \), we obtain that

\[ \Gamma(n+\frac{3}{2}) J_{n+1}(z) = \frac{1}{\sqrt{\pi}} \left( \frac{z}{2} \right)^{n+1} \int_{-1}^{1} e^{izt} (1-t^2)^{n+\frac{1}{2}} \, dt. \]

We divide this equation by \( n + \frac{1}{2} \) and integrate the integral in the right-hand side by parts. This gives

\[ \Gamma(n+\frac{1}{2}) J_{n+1}(z) = \frac{1}{\sqrt{\pi}} \left( \frac{z}{2} \right)^{n+1} \int_{-1}^{1} \frac{e^{izt}}{iz} (1-t^2)^{n-\frac{3}{2}} \, dt. \]

By combining (60) and (61) one obtains the given integral representations for \( J_n(z) + iJ_{n+1}(z) \) and \( J_n(z) - iJ_{n+1}(z) \).

NOTE. The functions \( J_n(z) \pm iJ_{n+1}(z) \) also play a role in asymptotics of orthogonal polynomials on the unit circle [17, 285]. The integral representations given in the previous theorem can also be found in [285].

COROLLARY 46. Except for \( z = 0 \), the functions \( J_n(z) \pm iJ_{n+1}(z) \) have no zeros on the imaginary axis.

PROOF. This follows immediately from the fact that the integrands (and thus also the integrals) in the representations given in the previous theorem are positive if \( z \) lies on the imaginary axis.

Equation (58) implies that the zeros of \( J_n(z) + iJ_{n+1}(z) \) are the reflections of the zeros of \( J_n(z) - iJ_{n+1}(z) \) about the real axis. Also, (58) and (59) imply that the zeros of \( J_n(z) \pm iJ_{n+1}(z) \) are symmetric with respect to the imaginary axis. Therefore, as far as the zeros are concerned, we may restrict our attention to only one of the functions \( J_n(z) \pm iJ_{n+1}(z) \), for example \( J_n(z) - iJ_{n+1}(z) \), and to the right half plane. The following theorem implies that it is even sufficient to consider \( J_n(z) - iJ_{n+1}(z) \) only in the fourth quadrant.

THEOREM 47. All the zeros of \( J_n(z) + iJ_{n+1}(z) \) lie in the upper half plane whereas all the zeros of \( J_n(z) - iJ_{n+1}(z) \) lie in the lower half plane.

PROOF. See Tadeppalli and Synolakis [266].

We already know that \( J_n(z) \pm iJ_{n+1}(z) \) has a zero of multiplicity \( n \) at the origin if \( n \geq 1 \). The following theorem tells us that all the other zeros are simple.
Theorem 48. The zeros of \( J_n(z) \pm iJ_{n+1}(z) \) that lie in \( \mathbb{C}_0 \) are simple.

Proof. Suppose that \( z^\ast \in \mathbb{C}_0 \) is a multiple zero of \( F_n(z) = J_n(z) - iJ_{n+1}(z) \). Then

\[
F_n(z^\ast) = J_n(z^\ast) - iJ_{n+1}(z^\ast) = 0
\]

and

\[
F'_n(z^\ast) = J'_n(z^\ast) - iJ'_{n+1}(z^\ast) = 0.
\]

By using the relations \([77, \S7.2.8]\)

\[
J_n(z) = \frac{n}{z}J_n(z) - J_{n+1}(z)
\]

\[
J'_{n+1}(z) = J_n(z) - \frac{n+1}{z}J_{n+1}(z)
\]

we obtain that

\[
F'_n(z^\ast) = (\frac{n}{z^\ast} - i)J_n(z^\ast) - (1 - \frac{n+1}{z^\ast})J_{n+1}(z^\ast) = 0.
\]

Equations (62) and (63) represent a homogeneous linear system of equations in \( J_n(z^\ast) \) and \( J_{n+1}(z^\ast) \). Its determinant is equal to \( i^{2n+1} \neq 0 \). It follows that \( J_n(z^\ast) = J_{n+1}(z^\ast) = 0 \). This implies that \( z^\ast \) must be real. Indeed, Bessel functions of the first kind and order \( > -1 \) have only real zeros, cf. Watson [295, \S15.27]. However, this is impossible as we have shown in Theorem 44 that \( F_n(z) \) has no zeros in \( \mathbb{R}_0 \). This proves that all the zeros of \( F_n(z) \) that lie in \( \mathbb{C}_0 \) are simple. The proof for \( J_n(z) + iJ_{n+1}(z) \) is analogous.

\[\square\]

3. The zeros of \([J_{n+1}(z)]^2 - J_n(z)J_{n+2}(z)\)

If \( n \geq 1 \), then \( J_n(z) \) has a zero of multiplicity \( n \) at the origin. The Bessel functions \( J_{n+1}(z) \) and \( J_{n+2}(z) \) have a zero of multiplicity \( n+1 \) respectively \( n+2 \) at the origin. Therefore, if \( n \geq 1 \), then \( G_n(z) = [J_{n+1}(z)]^2 - J_n(z)J_{n+2}(z) \) has a zero of multiplicity \( \geq 2n+2 \) at the origin. Using the Taylor series (56) one can easily verify that the multiplicity is in fact equal to \( 2n+2 \). This holds also in case \( n = 0 \). The inequality (cf. Szász [265])

\[
[J_{n+1}(z)]^2 - J_n(z)J_{n+2}(z) > \frac{[J_{n+1}(z)]^2}{n+2}, \quad z > 0,
\]

and equation (59) imply that no other zeros exist on the real axis. In Corollary 50 we will show that no other zeros exist on the imaginary axis.

If \( z^\ast \) is a zero of \( G_n(z) \), then (58) and (59) immediately imply that the same holds for \( \overline{z^\ast} \) and \(-z^\ast\). In other words, the zeros of \( G_n(z) \) are symmetric with respect to the real axis and about the origin. Thus we may restrict our attention to one quadrant in the complex plane, for example the first quadrant.
**Theorem 49.** \( G'_n(z) = \frac{2}{z}J_n(z)J_{n+2}(z) \) for all \( z \neq 0 \).

**Proof.** We use the recurrence relations [77, §7.2.8]

\[
\begin{align*}
J_n(z) + J_{n+2}(z) &= \frac{2}{z}(n+1)J_{n+1}(z) \\
J_n(z) - J_{n+2}(z) &= 2J'_{n+1}(z) \\
zJ'_{n+2}(z) + (n+2)J_{n+2}(z) &= zJ_{n+1}(z) \\
zJ'_n(z) - nJ_n(z) &= -zJ_{n+1}(z)
\end{align*}
\]

to eliminate \( J'_n(z) \), \( J_{n+1}(z) \), \( J'_{n+1}(z) \) and \( J'_{n+2}(z) \) from

\[
G'_n(z) = 2J_{n+1}(z)J'_{n+1}(z) - J_n(z)J'_{n+2}(z) - J'_n(z)J_{n+2}(z).
\]

It follows that

\[
G'_n(z) = \frac{2}{2(n+1)}(J_n(z) + J_{n+2}(z)) \left( \frac{1}{2}(J_n(z) - J_{n+2}(z)) \right)
\]

\[
- J_n(z)(J_{n+1}(z) - \frac{n+2}{z}J_{n+2}(z)) - J_{n+2}(z)(\frac{n}{z}J_{n+1}(z) - J_n(z))
\]

\[
= \frac{z}{2(n+1)}([J_n(z)]^2 - [J_{n+2}(z)]^2) - J_n(z)J_{n+1}(z) + \frac{n+2}{z}J_n(z)J_{n+2}(z)
\]

\[
- \frac{n}{z}J_n(z)J_{n+2}(z) + J_{n+1}(z)J_{n+2}(z)
\]

\[
= \frac{z}{2(n+1)}([J_n(z)]^2 - [J_{n+2}(z)]^2) - J_n(z)\left( \frac{z}{2(n+1)}(J_n(z) + J_{n+2}(z)) \right)
\]

\[
+ \frac{2}{z}J_n(z)J_{n+2}(z) + \frac{z}{2(n+1)}(J_n(z) + J_{n+2}(z))J_{n+2}(z)
\]

\[
= \frac{2}{z}J_n(z)J_{n+2}(z).
\]

This proves the theorem. \( \square \)

**Corollary 50.** Except for \( z = 0 \), the function \( [J_{n+1}(z)]^2 - J_n(z)J_{n+2}(z) \) has no zeros on the imaginary axis.

**Proof.** Define \( f_n(\zeta) : \mathbb{R} \rightarrow \mathbb{R} : \zeta \mapsto G_n(i\zeta) \). Then

\[
f'_n(\zeta) = iG'_n(i\zeta) = \frac{2}{\zeta}J_n(i\zeta)J_{n+2}(i\zeta).
\]

Let \( I_n(z) \) denote the modified Bessel function of the first kind of order \( n \). From the Taylor series

\[
I_n(z) = \left( \frac{z}{2} \right)^n \sum_{k=0}^{\infty} \frac{1}{k!(n+k)!} \left( \frac{z}{2} \right)^{2k}, \quad |z| < \infty,
\]

it immediately follows that \( J_n(z) = i^nI_n(-iz) \). Thus \( J_n(i\zeta) = i^nI_n(\zeta) \) and

\[
f'_n(\zeta) = (-1)^{n+1}\frac{2}{\zeta}J_n(\zeta)I_{n+2}(\zeta).
\]

Now we have to distinguish between \( \zeta > 0 \) and \( \zeta < 0 \).

- If \( \zeta > 0 \), then \( J_n(\zeta) > 0 \) and thus \( \text{sign} \ f'_n(\zeta) = (-1)^{n+1} \) independently of \( \zeta \). Thus \( f_n(\zeta) \) is either strictly increasing or strictly decreasing in \((0, +\infty)\). As \( f_n(0) = 0 \), it follows that \( f_n(\zeta) \) has no zeros in \((0, +\infty)\).
• If \( \zeta < 0 \) then, as \( I_n(-\zeta) = (-1)^n I_n(\zeta) \), it follows that \( \text{sign} I_n(\zeta) = (-1)^n \) and thus \( f_n(\zeta) = (-1)^{n+1}(-1)^n(-1)^{n+2} = (-1)^{3n+4} \), which is a gain independent of \( \zeta \). Thus \( f_n(\zeta) \) has no zeros in \((-\infty,0)\).

This proves the theorem. Note that \( f_n'(\zeta) \) has opposite sign in \((-\infty,0)\) and \((0,\infty)\). This is in accordance with the fact that \( f_n(\zeta) \) has a multiple zero of even multiplicity at \( \zeta = 0 \).

\[ \text{Corollary 51. } \text{The zeros of } [J_{n+1}(z)]^2 - J_n(z)J_{n+2}(z) \text{ that lie in } \mathbb{C}_0 \text{ are simple.} \]

\[ \text{Proof. } \text{Suppose that } z^* \in \mathbb{C}_0 \text{ is a multiple zero of } G_n(z), \]

\[ G_n(z^*) = G_n'(z^*) = 0. \]

Then Theorem 49 implies that \( J_n(z^*) = 0 \) or \( J_{n+2}(z^*) = 0 \). However, this cannot hold as we have proven that \( G_n(z) \) has no real zeros except at \( z = 0 \) whereas Bessel functions of the first kind and order \( > -1 \) have only real zeros, cf. Watson [295, §15.27].

Another argument goes as follows. If \( G_n'(z^*) = 0 \), then we have just seen that \( z^* \) must be real. Also, \( G_n(z^*) = 0 \) together with \( J_n(z^*) = 0 \) or \( J_{n+2}(z^*) = 0 \) implies that \( J_{n+1}(z^*) = 0 \). This is impossible because of the interlacing property of the zeros of Bessel functions of the first kind and of successive nonnegative integer orders, cf. Watson [295, §15.22].

4. Numerical results

The package ZEBEC was written for the Bessel functions \( J_n(z), Y_n(z), H^{(1)}_n(z), H^{(2)}_n(z) \) and their first derivatives, but it can be easily extended to calculate zeros of any analytic function \( f(z) \) whose zeros are known to be simple, provided that a Fortran 77 routine exists to evaluate the logarithmic derivative \( f'(z)/f(z) \). In Section 2 and 3 we have shown that all the zeros of \( F_n(z) = J_n(z) - iJ_{n+1}(z) \) and \( G_n(z) = [J_{n+1}(z)]^2 - J_n(z)J_{n+2}(z) \) that lie in \( \mathbb{C}_0 \) are simple. The zeros of these functions can therefore be calculated via ZEBEC. As

\[
\frac{F_n'(z)}{F_n(z)} = \frac{J_n'(z) - iJ_{n+1}'(z)}{J_n(z) - iJ_{n+1}(z)} \quad \text{and} \quad \frac{G_n'(z)}{G_n(z)} = \frac{\frac{2}{z}J_n(z)J_{n+2}(z)}{[J_{n+1}(z)]^2 - J_n(z)J_{n+2}(z)},
\]

it seems that one has to evaluate \( J_n(z), J_n'(z), J_{n+1}(z), J_{n+1}'(z) \) and \( J_n(z), J_{n+1}(z), J_{n+2}(z) \) to compute \( F_n'(z)/F_n(z) \) and \( G_n'(z)/G_n(z) \), resp. However, the recurrence relations [77, §7.2.8]

\[
J_n'(z) = \frac{n}{z}J_n(z) - J_{n+1}(z)
\]

\[
J_{n+1}'(z) = J_n(z) - \frac{n+1}{z}J_{n+1}(z)
\]

and

\[
J_{n+2}(z) = \frac{2(n+1)}{z}J_n(z) - J_{n+1}(z),
\]

respectively, imply that it is sufficient to compute \( J_n(z) \) and \( J_{n+1}(z) \).

We have calculated the first 30 zeros of \( J_n(z) - iJ_0(z) \) that lie in the fourth quadrant. The results are shown in Table 1. The reader may compare these with the values given by MacDonald [206, Table 7 on page 633] and conclude that our approximations are more accurate, in particular the approximations for the smaller zeros.
Table 1. Approximations for the first 30 zeros of $J_5(z) - iJ_6(z)$ that lie in the fourth quadrant.

Acknowledgements

The results in this chapter were obtained in collaboration with Pierre Verlinden.

I would like to thank Michael Vrahatis for providing me with references [204], [206] and [266], which initiated this research. I would also like to thank Mourad Ismail, Duncan MacDonald and Costas Synolakis for interesting discussions.
Part 3
Newton’s method for multiple zeros

We propose a modification of Newton’s method for computing multiple zeros of analytic mappings (in other words, multiple roots of systems of analytic equations). Under mild assumptions the iteration converges quadratically. It involves certain constants whose product is a lower bound for the multiplicity of the zero. As these constants are usually not known in advance, we devise an iteration in which not only an approximation for the zero is refined, but also approximations for these constants. Numerical examples illustrate the effectiveness of our approach.

This chapter corresponds to our paper [184].

1. Introduction

Consider a smooth function \( f : \mathbb{C} \to \mathbb{C} \) that has a zero of multiplicity \( \mu \) at the point \( z^* \). If \( \mu = 1 \), then Newton’s method converges quadratically to \( z^* \) if the initial iterate is sufficiently close to \( z^* \). If \( \mu > 1 \), then the convergence is only linear. In the latter case, if \( \mu \) is known in advance, quadratic convergence can be regained by considering the iteration

\[
(z^{(p+1)}) = (z^{(p)}) - \frac{\mu f(z^{(p)})}{f'(z^{(p)})}, \quad p = 0, 1, 2, \ldots
\]

Van de Vel [281, 283] devised an iteration in which not only an approximation for the zero is refined, but also an estimate of its multiplicity. King [177] analysed Van de Vel’s method and proved that its order of convergence is 1.554. He rearranged the order of the calculations and obtained an iteration that has order of convergence 1.618. This iteration proceeds as follows:

\[
\begin{cases}
\mu^{(p+1)} = \frac{u(z^{(p)})}{u(z^{(p)}) - u(z^{(p+1)})} \mu^{(p)} \\
(z^{(p+2)}) = (z^{(p+1)}) - \mu^{(p+1)} u(z^{(p+1)})
\end{cases}
\]

for \( p = 0, 1, 2, \ldots \), with initial \( z^{(0)} \) and \( \mu^{(0)} \), and after one preliminary quasi-Newton step \( z^{(1)} = z^{(0)} - \mu^{(0)} u(z^{(0)}) \), and where \( u(z) := f(z)/f'(z) \).

We generalize these two results to the multivariate case. We consider systems of analytic equations, i.e., analytic mappings \( f : \mathbb{C}^n \to \mathbb{C}^n \). The multidimensional version of (64) is formulated in Theorem 56. Instead of \( \mu \) the iteration now involves a diagonal matrix containing certain constants \( k_1, \ldots, k_n \), that are called orders. The product of these orders is a lower bound for \( \mu \) (Theorem 55). In Section 4 we present an algorithm in which an approximation for the zero as well as approximations for the
orders are refined iteratively. This iteration is our multidimensional generalization of (63). A lot of numerical examples illustrate our results.

At a multiple zero the Jacobian matrix of \( f \) is singular. The set of points \( z \in \mathbb{C}^n \) such that \( \det f'(z) = 0 \) is a codimension one smooth manifold through the zero. As soon as an iterate lies on this manifold, the iteration breaks down. We assume throughout this chapter that this does not happen. In other words, we assume that the initial iterate is such that the iteration is well defined at every step. This assumption enables us to focus entirely on the order of convergence.

The behaviour of Newton’s method in case the Jacobian is singular at the zero has been analysed extensively in the literature [65, 66, 67, 68, 69, 119, 120, 121, 122, 173, 174, 223, 245, 246, 309]. Many sufficient conditions for its convergence have been formulated. Under certain regularity and smoothness assumptions, the existence of special regions (cones, starlike regions) about the zero \( z^* \) has been proven. The Jacobian is regular in every point of these regions except in \( z^* \). If the initial iterate lies in such a region, then the Newton iterates will remain in this region and converge (linearly) to \( z^* \). We have not investigated the existence of such regions for the iterations presented in this chapter. Indeed, the proofs for the classical cases are extremely complex and it is not a priori clear how to extend them to our iterations.

In [68] a modification of Newton’s method is proposed that produces a sequence \( \{z^{(p)}\}_{p \geq 0} \) such that the subsequence \( \{z^{(2p)}\}_{p \geq 0} \) converges quadratically to the zero. However, this method works only in case the dimension of the null space of the Jacobian at the zero is equal to 1 or 2, and the projector onto this null space is known explicitly. Other modifications of Newton’s method have been proposed in [65, 174]. These methods result in superlinear or quadratic convergence but again require rather restrictive hypotheses to be satisfied and need additional information (certain constants, projectors, …) that is usually not available.

In [225, 226, 227] a ‘deflation algorithm’ was proposed for computing multiple roots of systems of nonlinear algebraic equations. The system to be solved is replaced by another one having the same root but with a lower multiplicity. While the deflation algorithm proceeds, the multiplicity is systematically reduced until it is equal to one and classical methods can be applied. However, this algorithm requires symbolic calculation and works only for systems of algebraic equations.

Other approaches that have been proposed include bordering methods [146, 147, 148, 195, 210], enlargement methods [224, 270, 296] and homotopy continuation methods [215, 216, 217].

All these methods require deciding whether the problem is singular: one should know in advance that the zero is multiple. This probably makes these methods unsuitable for general purpose use. As we will illustrate in Example 27, our method also works in case the zero is simple. Moreover, it requires no additional information, works under mild assumptions, and provides a lower bound for the multiplicity of the zero.
2. Preliminaries and notation

Let \( f = f(z) : \mathbb{C}^n \to \mathbb{C}^n \) be an analytic mapping, with \( z = (z_1, \ldots, z_n) \) and \( f = (f_1, \ldots, f_n) \). A point \( z^* \in \mathbb{C}^n \) is called a zero of \( f \) if \( f(z^*) = 0 \). An isolated zero \( z^* \) of \( f \) is called simple if the Jacobian matrix of \( f \) at \( z^* \) is regular, \( \det f'(z^*) \neq 0 \).

The following material is taken from the well-known book by Aizenberg and Yuzhakov [3].

**Proposition 52.** If the closure of a neighbourhood \( U_{z^*} \) of a zero \( z^* \) of \( f \) does not contain other zeros of \( f \), then there exists an \( \varepsilon > 0 \) such that for almost all \( \zeta \in \mathbb{C}^n, \|\zeta\| < \varepsilon \), the mapping

\[
z \mapsto f(z) - \zeta
\]

has only simple zeros in \( U_{z^*} \) and their number depends neither on \( \zeta \) nor on the choice of the neighbourhood \( U_{z^*} \).

The number of zeros of the mapping (66) in \( U_{z^*} \), indicated in this proposition is called the multiplicity of the zero \( z^* \) of \( f \) and is denoted by \( \mu_{z^*}(f) \). In other words, the multiplicity of an isolated zero of an analytic mapping is given by the number of simple zeros into which this zero desintegrates under a sufficiently small perturbation of the mapping.

The next result follows from the local invertibility of an analytic mapping at points where its Jacobian matrix is regular.

**Proposition 53.** The multiplicity of a simple zero is equal to 1.

**Proposition 54.** If \( z^* \) is an isolated zero of \( f \) and \( \det f'(z^*) = 0 \), then its multiplicity \( \mu_{z^*}(f) \) is larger than 1.

This statement justifies calling an isolated zero \( z^* \) of \( f \) multiple if \( \det f'(z^*) = 0 \). Now let \( z^* = (z_1^*, \ldots, z_n^*) \) be an isolated zero of \( f = (f_1, \ldots, f_n) \) such that

\[
f_j(z) = \sum_{|\alpha| \geq k_j} c_{j, \alpha} (z - z^*)^\alpha
\]

for \( j = 1, \ldots, n \) where \( \alpha \) is a multi-index, \( \alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n, |\alpha| = \alpha_1 + \cdots + \alpha_n \) and \( (z - z^*)^\alpha = (z_1 - z_1^*)^{\alpha_1} \cdots (z_n - z_n^*)^{\alpha_n} \). We call \( k_j \) the order of \( z^* \) as a zero of \( f_j \). Define

\[
P_j(z) := \sum_{|\alpha| = k_j} c_{j, \alpha} (z - z^*)^\alpha
\]

for \( j = 1, \ldots, n \). The homogeneous polynomial mapping

\[
P = P(z) := (P_1(z), \ldots, P_n(z))
\]

is called the homogeneous principal part of \( f \) at \( z^* \). The following theorem by Tsikh and Yuzhakov relates the multiplicity \( \mu_{z^*}(f) \) to the orders \( k_1, \ldots, k_n \).

**Theorem 55.** The multiplicity of an isolated zero \( z^* \) of \( f \) is equal to the product of the orders of \( z^* \) as a zero of \( f_1, \ldots, f_n \) if and only if \( z^* \) is an isolated zero of the mapping (67). Moreover, the inequality \( \mu_{z^*}(f) \geq k_1 \cdots k_n \) always holds.
3. A modification of Newton's method

**Theorem 56.** Suppose that \( z^{(0)} \) is such that the iteration
\[
    z^{(p+1)} = z^{(p)} - [f'(z^{(p)})]^{-1} \text{diag}(h_1, \ldots, h_n)f(z^{(p)}),
\]
\( p = 0, 1, 2, \ldots \),
is well defined for every \( p \). If \( \det P'(z) \neq 0 \) and if \( z^{(0)} \) is sufficiently close to \( z^* \), then \( z^{(p)} \) converges quadratically to \( z^* \). If \( \det P'(z) = 0 \), then the convergence is only linear.

**Proof.** Define
\[
    \hat{f}_j(z) := \sum_{|\alpha| \geq k_j + 1} c_{j, \alpha} (z - z^*)^\alpha
\]
for \( j = 1, \ldots, n \). As
\[
    f_j(z) = P_j(z) + \hat{f}_j(z) = \sum_{|\alpha| = k_j} c_{j, \alpha} (z - z^*)^\alpha + \hat{f}_j(z)
\]
for \( j = 1, \ldots, n \), it follows that
\[
    \frac{\partial f_i}{\partial z_k}(z) = \sum_{|\alpha| = k_j} \alpha_k c_{j, \alpha} (z_1 - z_1^*)^\alpha_1 \cdots (z_k - z_k^*)^\alpha_k - 1 \cdots (z_n - z_n^*)^\alpha_n + \frac{\partial \hat{f}_i}{\partial z_k}(z)
\]
for \( j, k = 1, \ldots, n \). Let \( e^{(p)} = (e_1^{(p)}, \ldots, e_n^{(p)}) := z^{(p)} - z^* \). Then the iteration can be written as
\[
    f'(z^* + e^{(p)}) e^{(p+1)} = f'(z^* + e^{(p)}) e^{(p)} - \text{diag}(h_1, \ldots, h_n)f(z^* + e^{(p)}).
\]
The \( j \)th component of the vector appearing in the right-hand side of (69) is given by
\[
    g_j(e^{(p)}) := \sum_{k=1}^n \frac{\partial f_i}{\partial z_k}(z^* + e^{(p)}) e_{k, p} - k_j f_j(z^* + e^{(p)})
\]
\[
    = \sum_{|\alpha| = k_j} \alpha_k c_{j, \alpha} e_{1, p}^{\alpha_1} \cdots e_{n, p}^{\alpha_n} + \frac{\partial \hat{f}_j}{\partial z_k}(z^* + e^{(p)}) e_{k, p}
\]
\[
    - k_j \left[ \sum_{|\alpha| = k_j} c_{j, \alpha} e_{1, p}^{\alpha_1} \cdots e_{n, p}^{\alpha_n} + \hat{f}_j(z^* + e^{(p)}) \right]
\]
\[
    = \sum_{|\alpha| = k_j} (|\alpha| - k_j) c_{j, \alpha} [e^{(p)}]^{\alpha} + \sum_{k=1}^n \frac{\partial \hat{f}_j}{\partial z_k}(z^* + e^{(p)}) e_{k, p} - k_j \hat{f}_j(z^* + e^{(p)})
\]
\[
    = \sum_{k=1}^n \sum_{|\alpha| \geq k_j + 1} \alpha_k c_{j, \alpha} [e^{(p)}]^{\alpha} - k_j \sum_{|\alpha| \geq k_j + 1} c_{j, \alpha} [e^{(p)}]^{\alpha}
\]
\[
    = \sum_{|\alpha| \geq k_j + 1} (|\alpha| - k_j) c_{j, \alpha} [e^{(p)}]^{\alpha}
\]
\[
    = \sum_{|\alpha| = k_j} c_{j, \alpha} [e^{(p)}]^{\alpha} + \sum_{|\alpha| > k_j + 1} (|\alpha| - k_j) c_{j, \alpha} [e^{(p)}]^{\alpha}.
\]
It follows that \( |g_j(e^{(p)})| = O(|e^{(p)}|^{k_j + 1}) \) for \( j = 1, \ldots, n \).
One can easily verify that $\det P'(z)$ is a homogeneous polynomial in $z - z^*$. Now there are two possibilities: either all its coefficients are equal to zero, $\det P' \equiv 0$, or $\det P'$ has degree $\sum_{j=1}^{n} k_j - n$. By using (68) we can write $\det f'$ as a sum of $2^n$ determinants, including $\det P'$, and it follows readily that

\begin{equation}
\det f'(z^* + e^{(p)}) = \begin{cases}
O(\|e^{(p)}\| \sum_{j=1}^{n} k_j - n) & \text{if } \det P' \neq 0, \\
O(\|e^{(p)}\| \sum_{j=1}^{n} k_j - n + 1) & \text{if } \det P' \equiv 0.
\end{cases}
\end{equation}

By setting $g := (g_1, \ldots, g_n)$ we can write (69) as

$$f'(z^* + e^{(p)}) e^{(p+1)} = g(e^{(p)}).$$

Cramer’s rule implies that

\begin{equation}
e^{(p+1)}_{j} = \frac{1}{\det f'(z^* + e^{(p)})} \times \begin{vmatrix}
\frac{\partial f_1}{\partial z_1}(z^* + e^{(p)}) & \cdots & \frac{\partial f_1}{\partial z_n}(z^* + e^{(p)}) \\
\vdots & \ddots & \vdots \\
\frac{\partial f_n}{\partial z_1}(z^* + e^{(p)}) & \cdots & \frac{\partial f_n}{\partial z_n}(z^* + e^{(p)})
\end{vmatrix}
\end{equation}

for $j = 1, \ldots, n$. The denominator in the right-hand side of (71) was examined in (70). One can easily verify that the numerator is $O(\|e^{(p)}\|\alpha)$ with $\alpha = (\sum_{j=1}^{n} k_j - n) - (k_j - 1) + (k_j + 1) = \sum_{j=1}^{n} k_j - n + 2$. This proves the theorem.

Remark. As already mentioned, the problem of analysing the possible structure of the set of initial iterates $z^{(0)}$ that lie in a neighbourhood of $z^*$ and guarantee convergence of the iteration is of considerable difficulty. We therefore restrict our attention entirely to the order of convergence.

In the following examples we used Mathematica 2.2. All the calculations were done in multiple precision arithmetic.

Example 21. The mapping $f = (f_1, f_2) = (z_1 \sin z_1 + z_2^2, z_2 + z_1 \sin z_2)$ has an isolated zero at $z^* = (0,0)$. The orders are $k_1 = 2$ and $k_2 = 1$. The homogeneous principal part of $f$ at $z^*$ is given by $P = P(z_1, z_2) = (z_1^2, z_2)$. It follows that $z^*$ is an isolated zero of $P$ and thus, according to Theorem 55, the multiplicity of $z^*$ as a zero of $f$ is equal to $k_1 \cdot k_2 = 2$. The Jacobian matrix of $P$ is given by

$$P'(z_1, z_2) = \begin{bmatrix} 2z_1 & 0 \\ 0 & 1 \end{bmatrix}$$

and thus $\det P'(z_1, z_2) \neq 0$. Therefore the iteration of Theorem 56 will converge quadratically. Table 1 illustrates this. The initial iterate was $z^{(0)} := (0.2, 0.2)$.

Example 22. The mapping $f = (f_1, f_2) = (z_1 z_2 + (\sin z_1)^2 + z_2^2, \sin z_1 \sin z_2)$ has an isolated zero at $z^* = (0,0)$. The orders are $k_1 = 2$ and $k_2 = 2$. The homogeneous principal part of $f$ at $z^*$ is given by $P = P(z_1, z_2) = (z_1^2 + z_1 z_2, z_1 z_2)$. As $P(z_1, z_2) = 0$ if and only if $z_1 = 0$ and $z_2$ arbitrary, it follows that $z^*$ is not an
isolated zero of $P$ and thus, according to Theorem 55, the multiplicity of $z^*$ as a zero of $f$ is strictly larger than $k_1 \cdot k_2 = 4$. The Jacobian matrix of $P$ is given by

$$P'(z_1, z_2) = \begin{bmatrix} z_1 \\
2z_1 + z_2 \\
2z_2 \\
1 
\end{bmatrix}$$

and thus $\det P'(z_1, z_2) \neq 0$. Therefore the iteration of Theorem 56 will converge quadratically. Table 2 illustrates this. The initial iterate was $z^{(0)} := (0.2, 0.2)$.

**Example 23.** The mapping

$$f = (f_1, f_2) = (z_1 + z_2 + z_1^2 + z_2^2 + 2z_1^3 + (\sin z_1)^3, 2(z_1 + z_2)^3 + z_1^4)$$

has an isolated zero at $z^* = (0, 0)$. The orders are $k_1 = 1$ and $k_2 = 3$. The homogeneous principal part of $f$ at $z^*$ is given by

$$P = P(z_1, z_2) = (z_1 + z_2, 2(z_1 + z_2)^3).$$

As $P(z_1, z_2) = 0$ if and only if $z_2 = -z_1$, it follows that $z^*$ is not an isolated zero of $P$ and thus, according to Theorem 55, the multiplicity of $z^*$ as a zero of $f$ is strictly larger than $k_1 \cdot k_2 = 3$. The Jacobian matrix of $P$ is given by

$$P'(z_1, z_2) = \begin{bmatrix} 1 \\
6(z_1 + z_2)^2 \\
6(z_1 + z_2)^2 \\
1 
\end{bmatrix}$$

and thus $\det P'(z_1, z_2) \equiv 0$. Therefore the iteration of Theorem 56 will converge linearly. Table 3 illustrates this. The initial iterate was $z^{(0)} := (0.2, 0.2)$.
\begin{table}
\begin{tabular}{|c|c|c|c|}
\hline
\(p\) & \(-\log_{10} |z_1^{(p)}|\) & \(-\log_{10} |z_2^{(p)}|\) & \(-\log_{10} \sqrt{|z_1^{(p)}|^2 + |z_2^{(p)}|^2}\) \\
\hline
0 & 0.7 & 0.7 & 0.6 \\
1 & 0.3 & 0.2 & 0.1 \\
2 & 0.9 & 0.6 & 0.6 \\
3 & 1.0 & 0.9 & 0.8 \\
4 & 1.6 & 1.5 & 1.4 \\
5 & 2.2 & 2.2 & 2.0 \\
\vdots & \vdots & \vdots & \vdots \\
11 & 5.8 & 5.8 & 5.6 \\
12 & 6.4 & 6.4 & 6.2 \\
13 & 7.0 & 7.0 & 6.8 \\
14 & 7.6 & 7.6 & 7.4 \\
15 & 8.2 & 8.2 & 8.0 \\
\vdots & \vdots & \vdots & \vdots \\
\hline
\end{tabular}
\caption{Example 23.}
\end{table}

Remark. In the previous examples we did not consider the case that \(\det P'(z) \equiv 0\) and \(z^*\) is an isolated zero of \(P\). In fact, this situation cannot occur. Proposition 52 immediately implies that \(\det P'(z)\) cannot be identically equal to zero near \(z^*\) if \(z^*\) is an isolated zero of \(P\). Thus \(\det P'(z) \equiv 0\) implies that \(z^*\) is not an isolated zero of \(P\).

Let \(d_1, \ldots, d_n \in \mathbb{C}_0\). Consider the iteration
\begin{equation}
\tag{72}
z^{(p+1)} = z^{(p)} - [f'(z^{(p)})]^{-1} \text{diag}(d_1, \ldots, d_n) f(z^{(p)}), \quad p = 0, 1, 2, \ldots,
\end{equation}
or, equivalently,
\begin{equation}
\tag{73}
f' (z^* + e^{(p)}) e^{(p+1)} = f'(z^* + e^{(p)}) e^{(p)} - \text{diag}(d_1, \ldots, d_n) f(z^* + e^{(p)}),
\end{equation}
where \(e^{(p)} := z^{(p)} - z^*\) for \(p = 0, 1, 2, \ldots\). Let \(g_j(e^{(p)})\) be the \(j\)th component of the vector appearing in the right-hand side of (73). Using the same reasoning as in the proof of Theorem 56, one can easily show that
\begin{equation}
\tag{68}
g_j(e^{(p)}) = \sum_{|\alpha| \geq k_j} (|\alpha| - d_j) c_{j, \alpha} [e^{(p)}]^{\alpha}.
\end{equation}
It follows that \(|g_j(e^{(p)})| = \mathcal{O}([e^{(p)}]^{k_j})\) for \(j = 1, \ldots, n\) and thus the iteration (72) converges only linearly to \(z^*\) (if \(\det P'(z) \neq 0\) and if \(z^{(0)}\) is sufficiently close to \(z^*\)). The special choice \(d_1 = k_1, \ldots, d_n = k_n\) gives quadratic convergence. But of course, the orders \(k_1, \ldots, k_n\) are usually not known in advance.

4. Van de Vel’s method

The following proposition will help us to devise an iteration for the unknown orders \(k_1, \ldots, k_n\).

Proposition 57. Let \(e \in \mathbb{C}^n\). Then \(P'(z^* + e) e = \text{diag}(k_1, \ldots, k_n) P(z^* + e)\).
Proof. Suppose $e = (e_1, \ldots, e_n)$. Then the $j$th component of $P^j(z^* + e)e$ is given by
\[
\sum_{k=1}^{n} \frac{\partial P_j}{\partial z_k} (z^* + e) e_k = \sum_{|\alpha| = k_j} |\alpha| f, e^\alpha = k_j P_j (z^* + e)
\]
for $j = 1, \ldots, n$. This proves the proposition.

Let $K := \text{diag}(k_1, \ldots, k_n)$ and suppose $z^{(p)}$ is our current approximation to $z^*$. Then, by the previous proposition,
\[
P'(z^{(p)}) (z^{(p)} - z^*) = KP(z^{(p)}).
\]
The next iterate $z^{(p+1)}$ is defined as
\[
z^{(p+1)} := z^{(p)} - [f'(z^{(p)})]^{-1} D^{(p)} f(z^{(p)})
\]
where $D^{(p)} := \text{diag}(d_1^{(p)}, \ldots, d_n^{(p)})$ contains our current approximations to the orders $k_1, \ldots, k_n$. Then
\[
P'(z^{(p+1)}) (z^{(p+1)} - z^*) = KP(z^{(p+1)}).
\]
Suppose that the matrices $P'(z^{(p)})$ and $P'(z^{(p+1)})$ are regular. Then
\[
z^{(p)} - z^* = [P'(z^{(p)})]^{-1} KP(z^{(p)})
\]
and
\[
z^{(p+1)} - z^* = [P'(z^{(p+1)})]^{-1} KP(z^{(p+1)}).
\]
By subtracting (75) and (76), and using (74) we obtain that
\[
[P'(z^{(p)})]^{-1} KP(z^{(p)}) - [P'(z^{(p+1)})]^{-1} KP(z^{(p+1)}) = [f'(z^{(p)})]^{-1} D^{(p)} f(z^{(p)}).
\]
If we replace $P$ by $f$ then this relation will be satisfied only approximatively. We use the resulting equation to define our next approximation $D^{(p+1)}$ to $K$:
\[
D^{(p+1)} f(z^{(p)}) - f'(z^{(p)}) [f'(z^{(p+1)})]^{-1} D^{(p+1)} f(z^{(p+1)}) = D^{(p)} f(z^{(p)}).
\]
This equation is solved for the diagonal matrix $D^{(p+1)}$ in the following way. Let
\[
F(z) := \text{diag}(f_1(z), \ldots, f_n(z)), \quad d^{(p)} := \begin{bmatrix} d_1^{(p)} \\ \vdots \\ d_n^{(p)} \end{bmatrix}
\]
and define $d^{(p+1)}$ in a similar way. Obviously
\[
D^{(p)} f(z^{(p)}) = F(z^{(p)}) d^{(p)}, \quad D^{(p+1)} f(z^{(p+1)}) = F(z^{(p+1)}) d^{(p+1)}, \quad \text{etc.}
\]
Therefore
\[
[F(z^{(p)}) - f'(z^{(p)}) [f'(z^{(p+1)})]^{-1} F(z^{(p+1)})] d^{(p+1)} = F(z^{(p)}) d^{(p)}.
\]
This is the formula that we will use to calculate $d^{(p+1)}$ from $d^{(p)}$, $z^{(p)}$ and $z^{(p+1)}$. If we define the matrix-valued function
\[
U(z) := [f'(z)]^{-1} \text{diag}(f_1(z), \ldots, f_n(z))
\]
for every $z \in \mathbb{C}^n$ such that $f'(z)$ is regular, (77) can be written as
\[
[U(z^{(p)}) - U(z^{(p+1)})] d^{(p+1)} = U(z^{(p)}) d^{(p)}.
\]
This is a multidimensional version of the iteration formula that was discovered by Van de Vel [281, 283]. Note the diagonal matrix in the definition of $U(z)$. If $n = 1$ then $U(z) = f(z)/f'(z)$. In the multidimensional case it is tempting to consider the vector-valued function $[f'(z)]^{-1}f(z)$ but, as we have just found out, one should replace the vector $f(z)$ by its corresponding diagonal matrix, to obtain a matrix-valued function $U(z)$. Now the iteration (74) can be written as

$$z^{(p+1)} = z^{(p)} - U(z^{(p)})d^{(p)}.$$  

From the foregoing considerations we extract the following iterative procedure:

$$d^{(p+1)} = \left[ U(z^{(p)}) - U(z^{(p)}) - U(z^{(p)})d^{(p)} \right]^{-1}U(z^{(p)})d^{(p)}$$

for $p = 0, 1, 2, \ldots$, starting with initial estimates $z^{(0)}$ for the zero $z^*$ and $d^{(0)}$ for the orders $[k_1 \cdots k_n]^T$. This is our generalization of Van de Vel’s method. It is a two-point method with memory. An equivalent formulation is the following:

$$d^{(p+1)} = \left[ U(z^{(p)}) - U(z^{(p)}) - U(z^{(p)})d^{(p)} \right]^{-1}U(z^{(p)})d^{(p)}$$

for $p = 0, 1, 2, \ldots$. This leads to the following algorithm.

**Algorithm (two-point version).**

**input** $z^{(0)}, d^{(0)}$

for $p = 0, 1, 2, \ldots$

1. Solve $f'(z^{(p)})\Delta z^{(p)} = -\text{diag}(d^{(p)})f(z^{(p)})$

   $z^{(p+1/2)} \leftarrow z^{(p)} + \Delta z^{(p)}$

2. Solve $F(z^{(p)}) = f'(z^{(p)})[f'(z^{(p+1/2)})^{-1}F(z^{(p+1/2)})]d^{(p+1)} = F(z^{(p)})d^{(p)}$

3. Solve $f'(z^{(p+1/2)})\Delta z^{(p+1/2)} = -\text{diag}(d^{(p+1)})f(z^{(p+1/2)})$

   $z^{(p+1)} \leftarrow z^{(p+1/2)} + \Delta z^{(p+1/2)}$

This method can be improved by noting that after step 1 is completed the first time, there is no reason ever to return to it. Instead the estimate of the orders can be improved (step 2) before each and every further quasi-Newton step (step 3). Thus the improved iteration may be written as

$$d^{(p+1)} = \left[ U(z^{(p)}) - U(z^{(p)}) - U(z^{(p)})d^{(p)} \right]^{-1}U(z^{(p)})d^{(p)}$$

for $p = 0, 1, 2, \ldots$, with initial $z^{(0)}$ and $d^{(0)}$, and after one preliminary quasi-Newton step $z^{(1)} = z^{(0)} - U(z^{(0)})d^{(0)}$. This one-point method with memory corresponds to King’s improvement of Van de Vel’s method [177].

**Algorithm (one-point version).**

**input** $z^{(0)}, d^{(0)}$

Solve $f'(z^{(0)})\Delta z^{(0)} = -\text{diag}(d^{(0)})f(z^{(0)})$

$z^{(1)} \leftarrow z^{(0)} + \Delta z^{(0)}$

for $p = 0, 1, 2, \ldots$
The columns labelled \("\text{two-point version}\) contain \(\log_{10} \|k\|_2 - k_1 / k_1\) and \(\log_{10} \|k\|_2 - k_2 / k_2\), respectively. The columns labelled \("\text{one-point version}\) contain \(-\log_{10} \|z^*\|_2\) and \(-\log_{10} \|z^*\|_2\), respectively. The initial iterates were \(z(0) := (0,0,1)\) and \(d(0) := (1,0)\). The two-point version is superior. Intuitively this is not a surprise, of course.

**Example 25.** Next we reconsider the mapping of Example 22, but shifted to the point \(z^* = (1,3)\). In other words, suppose

\[
f = (f_1, f_2) = (uv + (\sin u)^2 + v^3, \sin u \sin v)
\]

where \(u = z_1 - 1\) and \(v = z_2 - 3\). Table 5 compares both versions of our algorithm. The columns labelled \("\text{two-point version}\) and \("\text{one-point version}\) are now related to the (componentwise or normwise) \(\text{relative errors}\). The initial iterates were \(z(0) := (1,1,3.1)\) and \(d(0) := (1,1)\). Again the one-point version is superior.

**Example 26.** The mapping

\[
f = (f_1, f_2, f_3) = \\
(u^2 + u^2 \sin v + u^3 \sin w, v + uv + v^2 + u^2 \sin u, w^2 + u^3 + vw \sin w + v^4 + u^5)
\]
where \( u = z_1 - 1 \), \( v = z_2 - 2 \) and \( w = z_3 - 3 \) has an isolated zero at \( z^* = (1, 2, 5) \). The orders are \( k_1 = 2 \), \( k_2 = 1 \) and \( k_3 = 2 \). The homogeneous principal part of \( f \) at \( z^* \) is given by \( P = P(z_1, z_2, z_3) = ((z_1 - 1)^2, z_2 - 2, (z_3 - 5)^2) \). It follows that \( z^* \) is an isolated zero of \( P \) and thus, according to Theorem 55, the multiplicity of \( z^* \) as a zero of \( f \) is equal to \( k_1 \cdot k_2 \cdot k_3 = 4 \). We have used the one-point version of our algorithm. Table 6 contains minus the logarithm with base 10 of the (componentwise or normwise) relative errors. The initial iterates were \( z^{(0)} = (1, 2, 2, 5, 2) \) and \( d^{(0)} = (1, 1, 1) \).

**Example 27.** In our last example we consider a mapping that has a simple zero. The mapping \( f = (f_1, f_2, f_3) = (u + w^2 + uv + sin u sin v + v^2, v + w + v^2 + vw + (sin u)^3 + v w^2, w + uv + w^2 + u^2 sin v + w^3) \) where \( u = z_1 - 1 \), \( v = z_2 - 2 \) and \( w = z_3 - 5 \) has an isolated zero at \( z^* = (1, 2, 5) \). The orders are \( k_1 = 1 \), \( k_2 = 1 \) and \( k_3 = 1 \). The homogeneous principal part of \( f \) at \( z^* \) is given by \( P = P(z_1, z_2, z_3) = (z_1 - 1, z_2 - 2, z_3 - 5) \). It follows that \( z^* \) is an isolated zero of \( P \).
and thus, according to Theorem 55, the multiplicity of \( z^* \) as a zero of \( f \) is equal to \( k_1 \cdot k_2 \cdot k_3 = 1 \). Therefore \( \det f'(z^*) \neq 0 \). The iteration of Theorem 56 reduces to the classical Newton’s method. Table 7 shows the performance of the one-point version of our algorithm. The initial iterates were \( z^{(0)} = (1.2, 2.2, 5.2) \) and \( d^{(0)} := (1, 1, 1) \).

**Remark.** As soon as the iterates \( d_1^{(p)}, \ldots, d_n^{(p)} \) are sufficiently close to integers, one can determine the correct values of the orders \( k_1, \ldots, k_n \) and use the iteration of Theorem 56, of course.

**Remark.** King [177] analysed Van de Vel’s method [281], rearranged the order of the calculations, and gave a convergence proof for both iterations. He proved that Van de Vel’s method has order of convergence 1.554, and that his modification has order of convergence 1.618. The previous examples indicate that our multidimensional generalizations of these methods have the same corresponding order of convergence. Unfortunately, we have been unable to generalize King’s proof to the multidimensional setting. The main problem in the multidimensional case is that matrices occur instead of scalars and therefore commutativity gets lost.

## 5. Conclusions

In this chapter we have presented two iterations for computing multiple zeros of analytic mappings and studied their order of convergence.

The iteration of Theorem 56 is a multidimensional generalization of iteration (64). It can be used if the orders, which are positive integers related to the multiplicity of the zero, are known in advance. Under certain assumptions this iteration converges quadratically.

Our generalization of Van de Vel’s iteration is based on (78). The algorithm requires no prior knowledge about the multiplicity of the zero, which may even be simple, and proceeds by iterating on the zero as well as on the orders. Our numerical experiments indicate that the one-point version has order of convergence 1.618.
Several challenging open problems remain:

- We gave only an informal derivation of our multidimensional generalization (78) of Van de Vel's iteration, and the question remains whether it is possible to generalize King's proof.
- Local convergence regions: what is the structure of the set of initial iterates $z^{(0)}$ that lie in a neighbourhood of the zero $z^*$ and guarantee convergence?
- In the one-dimensional case the (discrete) dynamics of Newton's method are studied in terms of Julia and Fatou sets, Siegel disks, etc. (See, for example, [25] or [38].) The beautiful fractal-like images that illustrate these results are well-known. A global convergence analysis for the iterations presented in this chapter, in particular with respect to the role played by the singular manifold \( \{ z \in \mathbb{C}^n : \det f'(z) = 0 \} \), is a very interesting (but very difficult!) challenge.
- Newton's method has been studied in terms of its continuous counterpart: a system of autonomous differential equations whose Euler discretization yields (the damped) Newton's method. (See, for example, [211], [231] or [272].) Is it possible to formulate a continuous-time version of the iterations presented in this chapter, in particular for our generalization of Van de Vel's method, and how do the regions of attraction of these dynamical systems relate to the convergence regions of the discrete methods?

Acknowledgements

The results in this chapter were obtained in collaboration with Ann Haegemans.

I would like to thank Andreas Griewank for pointing out a flaw in the proof of a previous version of Theorem 56. I also thank José Manuel Gutiérrez, Hugo Van de Vel and Pierre Verlinden for stimulating discussions.
Part 4
CHAPTER 8

Superfast rational interpolation

We have already encountered rational interpolation at the end of Chapter 2 where we have used it to locate clusters of zeros of analytic functions.

In Section 3 of Chapter 2 we have already given a brief overview of the main papers concerning rational interpolation. A more complete list includes the papers [13, 14, 15, 18, 20, 21, 23, 27, 28, 29, 30, 31, 50, 59, 60, 73, 82, 100, 101, 114, 115, 116, 117, 124, 125, 126, 127, 131, 168, 207, 212, 249, 251, 252, 257, 258, 260, 274, 275, 294, 297, 298, 299, 301, 302, 303, 304]. We will not discuss these algorithms. Instead, we will focus on the algorithm of Van Barel and Bultheel [274]. It forms the basis for the algorithms that we will present in this chapter. Most of the algorithms for solving rational interpolation problems are likely to suffer from instabilities when applied in floating point arithmetic. This holds also for the algorithm of Van Barel and Bultheel. Hence, stabilizing techniques are necessary. As we will see, the algorithm of Van Barel and Bultheel has the advantage that it can be stabilized quite easily. In Section 1 we will show how pivoting can be incorporated into the algorithm. In Section 2 we will digress slightly to present a matrix or ‘block’ version of the algorithm. In Section 3 we will use a divide and conquer approach to obtain a recursive algorithm that is stabilized by giving what we will call “difficult interpolation points” a special treatment, via iterative refinement based on a formula for the inverse of a coupled Vandermonde matrix and via downdating.

This chapter is closely related to the next chapter, in which we will present fast and superfast solvers for linear systems of equations that have Hankel or Toeplitz structure. The chapters can be read independently from each other but one can also proceed as follows. After one has read Section 1 of this chapter (on how to incorporate pivoting into the algorithm of Van Barel and Bultheel), one may go directly to Section 1 of the next chapter (concerning a fast Hankel solver). Then one may return to Section 2 of this chapter (the block version of the results presented in Section 1) and then read Section 2 of the next chapter (a fast block Hankel solver). Finally, one may read Section 3 of this chapter (a stabilized divide and conquer approach) and then continue with Sections 3 and 4 (a superfast Hankel solver and a superfast Toeplitz solver) of the next chapter.

Part of this chapter is contained in our papers [187], [189], [278] and [279].

1. The algorithm \texttt{RatInt}

Let $n$ be a positive integer. Suppose that the complex numbers $s_1, \ldots, s_{2n} \in \mathbb{C}$ as well as the complex column vectors $f_1, \ldots, f_{2n} \in \mathbb{C}^{2 \times 1}$ are given. Consider the
interpolation problem

\[(79) \quad f_k^T B(s_k) = [0 \ 0], \quad k = 1, \ldots, 2n,\]

where

\[B(z) = \begin{bmatrix} n_\ell(z) & n_r(z) \\ d_\ell(z) & d_r(z) \end{bmatrix} \in \mathbb{C}[z]^{2 \times 2}\]

with \(\deg n_\ell(z) = n\), \(\deg d_\ell(z) \leq n - 1\), \(\deg n_r(z) \leq n - 1\) and \(\deg d_r(z) = n\), and \(n_\ell(z)\) as well as \(d_r(z)\) monic, i.e., where \(B(z)\) is a monic \(2 \times 2\) matrix polynomial of degree \(n\). Let us assume that this interpolation problem has a unique solution \(B^*(z)\). In the next chapter we will encounter linearized rational interpolation problems that can easily be formulated in this way.

Algorithm \textsc{RatInt} calculates a \(2 \times 2\) matrix polynomial \(B(z)\) of degree \(n\) that satisfies (79) and whose highest degree coefficient \(A\) \(\in \mathbb{C}^{2 \times 2}\) is nonsingular. This implies that \(B(z) \equiv B^*(z) A\). We will show below that \(\det A = 1\). This algorithm is a straightforward adaptation of the algorithm given by Van Barel and Bultheel [274]. Therefore we state it without proof. Our version incorporates pivoting, which is an important advantage as it enhances the numerical stability.

Define the column vectors \(s, L_s, R_s \in \mathbb{C}^{n \times 1}\) as

\[
s := \begin{bmatrix} s_1 \\ \vdots \\ s_{2n} \end{bmatrix}, \quad L_s := \begin{bmatrix} f_1(1) \\ \vdots \\ f_{2n}(1) \end{bmatrix} \quad \text{and} \quad R_s := \begin{bmatrix} f_1(2) \\ \vdots \\ f_{2n}(2) \end{bmatrix}.
\]

function \([B(z)] \leftarrow \textsc{RatInt}(s, L_s, R_s, 2n)\)

\[
B(z) \leftarrow \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}
\]

status \(\leftarrow 1\)

- status = 1 : both the left and the right choice are possible
- status = 2 : only the left choice is possible
- status = 3 : only the right choice is possible
- flow = 1 : \(L_s\) contains the pivot element
- flow = 2 : \(R_s\) contains the pivot element

for \(j = 1 : 2n\)

select case (status)

case (1)

1.1a \(\max R, \operatorname{pivR} \leftarrow \max_{j \leq k \leq 2n} \{\max \{|\Re R_s(k)|, |\Im R_s(k)|\}\}\)

1.1b \(\max L_s, \operatorname{pivL} \leftarrow \max_{j \leq k \leq 2n} \{\max \{|\Re L_s(k)|, |\Im L_s(k)|\}\}\)

if \(\max R > \max L_s\) then

1.1c \(\operatorname{piv} \leftarrow \operatorname{pivR}; \text{flow} \leftarrow 2; \text{status} \leftarrow 2\)

else

1.1d \(\operatorname{piv} \leftarrow \operatorname{pivL}; \text{flow} \leftarrow 1; \text{status} \leftarrow 3\)
end if

case (2)

1.2a \(\max L_s, \operatorname{pivL} \leftarrow \max_{j \leq k \leq 2n} \{\max \{|\Re L_s(k)|, |\Im L_s(k)|\}\}\)

1.2b \(\operatorname{piv} \leftarrow \operatorname{pivL}; \text{flow} \leftarrow 1; \text{status} \leftarrow 1\)
\[ \text{case (3)} \]
\[
1.3a \ [\max R, \text{pivR}] \leftarrow \max_{j \leq k \leq 2n} \{ \max \{ |\text{Re} R_s(k)|, |\text{Im} R_s(k)| \} \}
\]
\[
1.3b \ \text{piv} \leftarrow \text{pivR}; \ \text{flow} \leftarrow 2; \ \text{status} \leftarrow 1
\]
\end{select}

\[ s(j) \leftarrow s(\text{piv}); \ L_s(j) \leftarrow L_s(\text{piv}); \ R_s(j) \leftarrow R_s(\text{piv}) \]
\begin{select case (flow)}
\end{select}

\[ \text{case (1)} \]
\[
2.1a \ \mu \leftarrow R_s(j)/L_s(j)
\]
\[
2.1b \ B(z) \leftarrow B(z) \begin{bmatrix} z - s_j & -\mu \\ 0 & 1 \end{bmatrix}
\]
\[
\text{for } k = j + 1 : 2n
\]
\[
2.1c \ L_s(k) \leftarrow (s_k - s_j)L_s(k)
\]
\[
2.1d \ R_s(k) \leftarrow -\mu L_s(k) + R_s(k)
\]
\end{for}

\[ \text{case (2)} \]
\[
2.2a \ \mu \leftarrow L_s(j)/R_s(j)
\]
\[
2.2b \ B(z) \leftarrow B(z) \begin{bmatrix} 1 & 0 \\ -\mu & z - s_j \end{bmatrix}
\]
\[
\text{for } k = j + 1 : 2n
\]
\[
2.2c \ L_s(k) \leftarrow L_s(k) - \mu R_s(k)
\]
\[
2.2d \ R_s(k) \leftarrow (s_k - s_j)R_s(k)
\]
\end{for}

\end{select}

\end{for}

We have written down steps 1.1a, 1.1b, 1.2a and 1.3a using standard Matlab notation: the maxima are computed together with their location.

If the algorithm first performs a “left” step (flow = 1) and then a “right” step (flow = 2), then the highest degree coefficient (h.d.c.) of \( B(z) \) is multiplied on the right by

\[
\text{h. d. c.} \begin{bmatrix} z - s & -\mu \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\mu' & z - s' \end{bmatrix} = \begin{bmatrix} 1 & -\mu \\ 0 & 1 \end{bmatrix}.
\]

If, on the other hand, the algorithm first performs a right step and then a left step, then the h.d.c. of \( B(z) \) is multiplied on the right by

\[
\text{h. d. c.} \begin{bmatrix} 1 & 0 \\ -\mu & z - s \end{bmatrix} \begin{bmatrix} z - s' & -\mu' \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\mu & 1 \end{bmatrix}.
\]

Together with the initialization of \( B(z) \) this implies that the determinant of the h.d.c of \( B(z) \) is equal to 1.

The algorithm requires \( O(n^2) \) flops. We postpone a more detailed discussion of the arithmetic complexity of \textsc{RatInt} to Subsection 1.3 of the next chapter.

The multipliers \( \mu \) that have to be computed in step 2.1a or 2.2a are well defined. Indeed, consider the set \( S \) of all the column vector polynomials \( w(z) \in \mathbb{C}[z]^{2 \times 1} \) that satisfy the interpolation conditions

\[
f_k^T w(s_k) = 0, \quad k = 1, \ldots, 2n.
\]
If \( w(z) \in \mathbb{C}[z]^{2 \times 1} \) is an arbitrary vector polynomial, then the left-hand side of (80) is called the residual with respect to \( w \) at the interpolation point \( s_k \). Every element of \( S \) is thus such that the residuals at all the interpolation points are equal to zero.

Note that initially, the vectors \( L_s \) and \( R_s \) contain the residuals with respect to the left and right column vector polynomials corresponding to the initialization of \( B(z) \). The algorithm proceeds by pivoting (to change the order of the interpolation points), which results in a corresponding change in the order of the entries of \( L_s \) and \( R_s \), by updating \( B(z) \) in such a way that the residuals at the selected interpolation point become zero and by updating the residuals at the remaining interpolation points.

If after the execution of step \( j < 2n \) all the residuals in \( R_s \) or \( L_s \) would be equal to zero, then the interpolation problem (80) would have a solution of degree \( < n \). This is impossible, as we will now show. The set \( S \) forms a submodule of the \( \mathbb{C}[z] \)-module \( \mathbb{C}[z]^{2 \times 1} \). A basis for \( S \) always consists of exactly two elements [275, Theorem 3.1]. Let \( \{B_1(z), B_2(z)\} \) be a basis for \( S \). Then every element \( w(z) \in S \) can be written in a unique way as \( w(z) = \alpha_1(z)B_1(z) + \alpha_2(z)B_2(z) \) with \( \alpha_1(z), \alpha_2(z) \in \mathbb{C}[z] \). The matrix polynomial \( B(z) : = \begin{bmatrix} B_1(z) & B_2(z) \end{bmatrix} \in \mathbb{C}[z]^{2 \times 2} \) is called a basis matrix. Basis matrices can be characterized as follows.

**Theorem 58.** A matrix polynomial \( C(z) = \begin{bmatrix} C_1(z) & C_2(z) \end{bmatrix} \in \mathbb{C}[z]^{2 \times 2} \) is a basis matrix if and only if \( C_1(z), C_2(z) \in S \) and \( \deg \det C(z) = 2n \).

**Proof.** This follows immediately from [275, Theorem 4.1]. □

Note that \( B^*(z) \) is a basis matrix.

A matrix polynomial is called column reduced if the highest degree coefficients of its column vector polynomials are linearly independent. Every basis matrix can be transformed into a column reduced basis matrix [275, p. 455]. Note that \( B^*(z) \) is column reduced.

**Theorem 59.** Let \( \delta_1 := \deg B_1(z) \) and \( \delta_2 := \deg B_2(z) \). If \( B(z) \) is column reduced, then every element \( w(z) \in S \) having degree \( \leq \delta \) can be written in a unique way as \( w(z) = \alpha_1(z)B_1(z) + \alpha_2(z)B_2(z) \) with \( \alpha_1(z) \in \mathbb{C}[z], \deg \alpha_1(z) \leq \delta - \delta_1 \) and \( \alpha_2(z) \in \mathbb{C}[z], \deg \alpha_2(z) \leq \delta - \delta_2 \).

**Proof.** See [275, Theorem 3.2]. □

**Corollary 60.** The interpolation problem (80) has no nontrivial solution of degree \( < n \).

**Proof.** \( B^*(z) \) is a column reduced basis matrix. Its column degrees are equal to \( n \). Suppose that \( \delta < n \). Then \( \deg \alpha_1(z) < 0 \) and \( \deg \alpha_2(z) < 0 \) and thus \( \alpha_1(z) \equiv 0 \) and \( \alpha_2(z) \equiv 0 \). This implies that \( w(z) \equiv [0 \ 0]^T \).

2. **The algorithms** \texttt{BlockRatInt}_{\lambda} \text{ and } \texttt{BlockRatInt}_{\rho}

We will now formulate a matrix or ‘block’ version of the results presented in the previous section.

Let \( n \) and \( p \) be positive integers. Let the complex numbers \( s_1, \ldots, s_{2n} \in \mathbb{C} \) and the block column vectors \( f_{\lambda, 1}, \ldots, f_{\lambda, 2n} \in \mathbb{C}^{2p \times p} \) be given. Consider the interpolation
where \( \deg N_\ell(z) = n, \deg D_\ell(z) \leq n - 1, \deg N_r(z) \leq n - 1 \) and \( \deg D_r(z) = n \), and \( N_\ell(z) \) as well as \( D_r(z) \) monic, i.e., where \( B_\lambda(z) \) is a monic \( 2p \times 2p \) block matrix polynomial of degree \( n \). Let us assume that this interpolation problem has a unique solution \( B_\lambda^*(z) \). Similarly, let the block column vectors \( f_{\rho,1}, \ldots, f_{\rho,2n} \in \mathbb{C}^{2p \times p} \) be given and consider the interpolation problem

\[
B_{\rho}(s_k)f_{\rho,k} = \begin{bmatrix} O_p \\ O_p \end{bmatrix}, \quad k = 1, \ldots, 2n,
\]

where

\[
B_{\rho}(z) = \begin{bmatrix} N_u(z) & D_u(z) \\ N_l(z) & D_l(z) \end{bmatrix} \in \mathbb{C}[z]^{2p \times 2p}
\]

where \( \deg N_u(z) = n, \deg D_u(z) \leq n - 1, \deg N_l(z) \leq n - 1 \) and \( \deg D_l(z) = n \), and \( N_u(z) \) as well as \( D_u(z) \) monic, i.e., where \( B_{\rho}(z) \) is a monic \( 2p \times 2p \) block matrix polynomial of degree \( n \). Let us assume that this interpolation problem has a unique solution \( B_{\rho}^*(z) \). In the next chapter we will encounter linearized rational interpolation problems that can easily be formulated in this way.

The algorithm BlockRatInt_\lambda calculates a \( 2p \times 2p \) matrix polynomial \( B_\lambda(z) \) of degree \( n \) that satisfies (81) and whose highest degree coefficient \( A_\lambda \in \mathbb{C}^{2p \times 2p} \) is nonsingular. This implies that \( B_\lambda(z) \equiv B_\lambda^*(z)A_\lambda \). Hence, to obtain \( B_\lambda^*(z) \) we have to multiply \( B_\lambda(z) \) to the right by the inverse of \( A_\lambda \). We leave it to the reader to formulate an analogous algorithm BlockRatInt_\rho for computing a \( 2p \times 2p \) block matrix polynomial \( B_\rho(z) \) of degree \( n \) that satisfies (82) and whose highest degree coefficient \( A_\rho \in \mathbb{C}^{2p \times 2p} \) is nonsingular. The latter implies that \( B_\rho(z) \equiv A_\rho B_\rho^*(z) \). Below we will prove that \( \det A_\lambda = \det A_\rho = 1 \). These algorithms are easy generalizations of the algorithm RatInt.

Define the column vectors \( s \) and Residuals as

\[
s := \begin{bmatrix} s_1 \\ \vdots \\ s_1 \\ \vdots \\ s_n \end{bmatrix} \quad \text{and} \quad \text{Residuals} := \begin{bmatrix} f_{\lambda,1} \\ \vdots \\ f_{\lambda,2n} \end{bmatrix}.
\]

function \([B_\lambda(z)] \gets \text{BlockRatInt}_\lambda(s, \text{Residuals}, 2n, p)\)

\[
B_\lambda(z) \leftarrow \begin{bmatrix} I_p & 0_p \\ 0_p & I_p \end{bmatrix}
\]
\[ C \leftarrow \{1, \ldots, 2p\} \]

for \( j = 1 : 2pn \)
1. Determine \( \text{piv} \in \{j, \ldots, 2pn\} \) and \( \text{col} \in C \) such that
\[
\max\{|\text{Re Residuals}(\text{piv}, \text{col})|, |\text{Im Residuals}(\text{piv}, \text{col})|\}
= \max_{j \leq k \leq 2pn} \{\max\{|\text{Re Residuals}(k, l)|, |\text{Im Residuals}(k, l)|\}\}.
\]
2. \( s(j) \leftrightarrow s(\text{piv}); \text{Residuals}(j, :) \leftrightarrow \text{Residuals}(\text{piv}, :) \)
3. for \( l \in \{1, \ldots, 2p\} \setminus \{\text{col}\} \)
   3.1 \( \mu(l) \leftarrow \text{Residuals}(j,l)/\text{Residuals}(j,\text{col}) \)
end for
3.2 \( B_\lambda(z) \leftarrow B_\lambda(z) \times \begin{bmatrix} 1 \\ \vdots \\ -\mu(1) & \cdots & -\mu(\text{col} - 1) & z - s(j) & -\mu(\text{col} + 1) & \cdots & -\mu(2p) \\ \vdots & & & & & & \vdots & \end{bmatrix} \)
for \( k = j + 1 : 2pn \)
   for \( l \in \{1, \ldots, 2p\} \setminus \{\text{col}\} \)
   3.3a \( \text{Residuals}(k,l) \leftarrow \text{Residuals}(k,l) - \mu(l)\text{Residuals}(k,\text{col}) \)
end for
   3.3b \( \text{Residuals}(k,\text{col}) \leftarrow (s(k) - s(j))\text{Residuals}(k,\text{col}) \)
end for
4. \( C \leftarrow C \setminus \{\text{col}\}; \text{if } C = \emptyset \text{ then } C \leftarrow \{1, \ldots, 2p\} \)
end for

Step 3.2 implies that \( \det B_\lambda(z) \leftarrow \det B_\lambda(z)(z - s(j)) \) for \( j = 1 : 2pn \). Thus
\[
\det B_\lambda(z) \equiv (z - s_1)^p \cdots (z - s_{2n})^p,
\]
a monic polynomial of degree \( 2pn \). As \( \det B_\lambda(z) \equiv \det B_\lambda^*(z) \det A_\lambda \) and \( \det B_\lambda^*(z) \) is also a monic polynomial of degree \( 2pn \), we may conclude that \( \det A_\lambda = 1 \). A similar argument shows that \( \det A_\rho = 1 \).

The algorithms BlockRatInt and BlockRatInt each require \( \mathcal{O}(p^3n^2) \) flops. We postpone a more detailed discussion of the arithmetic complexity to Subsection 2.3 of the next chapter.

The multipliers \( \mu(l) \) that have to be computed in step 3.1 are well defined. Indeed, consider the set \( \mathcal{S} \) of all the column vector polynomials \( w(z) \in \mathbb{C}[z]^{2p \times 1} \) that satisfy the interpolation conditions
\[
(p) \quad f_{\lambda,k}^* w(s_k) = O_{p \times 1}, \quad k = 1, \ldots, 2n.
\]
If \( w(z) \in \mathbb{C}[z]^{2p \times 1} \) is an arbitrary vector polynomial, then the left-hand side of (83) is called the residual with respect to \( w \) at the interpolation point \( s_k \). Every element of \( \mathcal{S} \) is thus such that the residuals at all the interpolation points are equal to zero.
If after the execution of step \( j < 2p \) all the residuals would be equal to zero, then the interpolation problem (83) would have a solution of degree \( < n \). This is impossible, as we will now show. The set \( \mathcal{S} \) forms a submodule of the \( \mathbb{C}[z] \)-module \( \mathbb{C}[z]^{2p \times 1} \). A basis for \( \mathcal{S} \) always consists of exactly \( 2p \) elements, i.e., the dimension of \( \mathcal{S} \) is equal to \( 2p \) [275, Theorem 3.1]. Let \( \{B_k(z)\}_{k=1}^{2p} \) be a basis for \( \mathcal{S} \). Then every element \( w(z) \in \mathcal{S} \) can be written in a unique way as \( w(z) = \sum_{k=1}^{2p} \alpha_k(z)B_k(z) \) with \( \alpha_k(z) \in \mathbb{C}[z] \) for \( k = 1, \ldots, 2p \). The matrix polynomial \( B(z) := [B_1(z) \ B_2(z) \ \cdots \ B_{2p}(z)] \in \mathbb{C}[z]^{2p \times 2p} \) is called a basis matrix. Basis matrices can be characterized as follows.

**Theorem 61.** A matrix polynomial \( C(z) = [C_1(z) \ C_2(z) \ \cdots \ C_{2p}(z)] \in \mathbb{C}[z]^{2p \times 2p} \) is a basis matrix if and only if \( C_k(z) \in \mathcal{S} \) for \( k = 1, \ldots, 2p \) and \( \deg \det C(z) = 2p \).

**Proof.** This follows immediately from [275, Theorem 4.1]. \( \square \)

Note that \( B_1(z) \) is a basis matrix.

A matrix polynomial is called column reduced if the highest degree coefficients of its column vector polynomials are linearly independent. Every basis matrix can be transformed into a column reduced basis matrix [275, p. 455]. Note that \( B_1(z) \) is column reduced.

**Theorem 62.** Let \( \delta_k := \deg B_k(z) \) for \( k = 1, \ldots, 2p \). If \( B(z) \) is column reduced, then every element \( w(z) \in \mathcal{S} \) having degree \( \leq \delta \) can be written in a unique way as \( w(z) = \sum_{k=1}^{2p} \alpha_k(z)B_k(z) \) with \( \alpha_k(z) \in \mathbb{C}[z] \) and \( \deg \alpha_k(z) \leq \delta - \delta_k \) for \( k = 1, \ldots, 2p \).

**Proof.** See [275, Theorem 3.2]. \( \square \)

**Corollary 63.** The interpolation problem (83) has no nontrivial solution of degree \( < n \).

**Proof.** \( B_1(z) \) is a column reduced basis matrix. Its column degrees are equal to \( n \). Suppose that \( \delta < n \). Then \( \deg \alpha_k(z) < 0 \) and thus \( \alpha_k(z) \equiv 0 \) for \( k = 1, \ldots, 2p \). This implies that \( w(z) \equiv O_{2p \times 1} \). \( \square \)

3. A stabilized divide and conquer approach

We will now return to the setting of Section 1. Recall that the set \( \mathcal{S} \) has been defined as the set of all the column vector polynomials \( w(z) \in \mathbb{C}[z]^{2 \times 1} \) that satisfy the interpolation conditions

\[
f_j^T w(s_k) = 0, \quad k = 1, \ldots, 2n.
\]

Within the submodule \( \mathcal{S} \) we want to be able to consider solutions \( w(z) \) that satisfy additional conditions concerning their degree-structure. To describe the degree-structure of a vector polynomial, we use the concept of \( \tau \)-degree [275]. Let \( \tau \in \mathbb{Z} \). The \( \tau \)-degree of a vector polynomial \( w(z) = [w_1(z) \ w_2(z)]^T \in \mathbb{C}[z]^{2 \times 1} \) is defined as a generalization of the classical degree:

\[
\tau \text{-deg } w(z) := \max \{ \deg w_1(z), \deg w_2(z) - \tau \}
\]

with \( \tau \text{-deg } 0 := -\infty \). The \( \tau \)-highest degree coefficient of a vector polynomial \( [w_1(z) \ w_2(z)]^T \) with \( \tau \)-degree \( \delta \) is defined as the vector \( [\omega_1 \ \omega_2]^T \) with \( \omega_1 \) the coefficient of \( z^\delta \) in \( w_1(z) \) and \( \omega_2 \) the coefficient of \( z^{\delta+\tau} \) in \( w_2(z) \). A set of vector
polynomials in \( \mathbb{C}[z]^{2 \times 1} \) is called \( \tau\)-reduced if the \( \tau\)-highest degree coefficients are linearly independent. Every basis of \( \mathcal{S} \) can be transformed into a \( \tau\)-reduced one. For details, we refer to [275]. Once we have a basis in \( \tau\)-reduced form, the elements of \( \mathcal{S} \) can be parametrized as follows.

**Theorem 64.** Let \( \{B_1(z), B_2(z)\} \) be a \( \tau\)-reduced basis for \( \mathcal{S} \). Define \( \delta_1 := \tau\deg B_1(z) \) and \( \delta_2 := \tau\deg B_2(z) \). Then every element \( w(z) \in \mathcal{S} \) having \( \tau\)-degree \( \leq \delta \) can be written in a unique way as

\[
w(z) = \alpha_1(z)B_1(z) + \alpha_2(z)B_2(z)
\]

with \( \alpha_1(z), \alpha_2(z) \in \mathbb{C}[z] \), \( \deg \alpha_1(z) \leq \delta - \delta_1 \) and \( \deg \alpha_2(z) \leq \delta - \delta_2 \).

**Proof.** See Van Barel and Bultheel [275, Theorem 3.2].

The following theorem will enable us to devise an interpolation algorithm that is based on a divide and conquer approach. It shows how basis matrices can be coupled in case the degree-structure is important.

**Theorem 65.** Suppose \( K \) is a positive integer. Let \( \sigma_1, \ldots, \sigma_K \in \mathbb{C} \) be mutually distinct and let \( \phi_1, \ldots, \phi_K \in \mathbb{C}^{2 \times 1} \). Suppose that \( \phi_k \neq [0 \ 0]^T \) for \( k = 1, \ldots, K \). Let \( 1 \leq \kappa \leq K \). Let \( \tau_\kappa \in \mathbb{Z} \). Suppose that \( B_\kappa(z) \in \mathbb{C}[z]^{2 \times 2} \) is a \( \tau_\kappa\)-reduced basis matrix with basis vectors having \( \tau_\kappa\)-degree \( \delta_1 \) and \( \delta_2 \) respectively, corresponding to the interpolation data

\[
\{ (\sigma_k, \phi_k) : k = 1, \ldots, \kappa \}.
\]

Let \( \tau_{\kappa \rightarrow K} := \delta_1 - \delta_2 \). Let \( B_{\kappa \rightarrow K}(z) \in \mathbb{C}[z]^{2 \times 2} \) be a \( \tau_{\kappa \rightarrow K}\)-reduced basis matrix corresponding to the interpolation data

\[
\{ (\sigma_k, \phi_k^T \sigma_k \phi_k) : k = \kappa + 1, \ldots, K \}.
\]

Then \( B_K(z) := B_\kappa(z)B_{\kappa \rightarrow K}(z) \) is a \( \tau_K\)-reduced basis matrix corresponding to the interpolation data

\[
\{ (\sigma_k, \phi_k) : k = 1, \ldots, K \}.
\]

**Proof.** See Van Barel and Bultheel [276, Theorem 3].

The following algorithm implements this theorem. We start with \( N_\kappa \geq 2 \) interpolation points, where \( N_\kappa \) is a power of 2. These interpolation points are split into two equal parts. As \( N_\kappa \) is a power of 2, this process can be repeated. At the lowest level the interpolation problems are solved by our fast solver \textsc{RatInt}.

**Recursive function** \( [B(t)] \leftarrow \textsc{RecRatInt}(s, L_s, R_s, N_s, \tau) \)

-- \( \tau \in \mathbb{Z} \)
-- \( N_s = 2^{p+1} \) for some \( p \in \mathbb{N} \) the number of interpolation conditions
-- \( s \in \mathbb{C}^{N_s \times 1} \): the (mutually distinct) interpolation points
-- \( L_s, R_s \in \mathbb{C}^{N_s \times 1} \): the initial left and right residual vectors
-- \( B(t) \in \mathbb{C}[t]^{2 \times 2} \): a \( \tau\)-reduced basis matrix corresponding to the given interpolation data

\[
\text{if } N_s > 2^\text{limit} \text{ then }
\]

\[
[s_1, L_{s_1}, R_{s_1}, s_2, L_{s_2}, R_{s_2}] \leftarrow \textsc{Split}(s, L_s, R_s)
\]

\[
[B(t)] \leftarrow \textsc{RecRatInt}(s_1, L_{s_1}, R_{s_1}, N_s/2, \tau)
\]

\[
\text{for } k = 1(1)N_s/2
\]
\[
[\hat{L}_{s_2}(k) \quad \hat{R}_{s_2}(k)] \leftarrow [L_{s_2}(k) \quad R_{s_2}(k)] \cdot B_1(s_2(k))
\]
end for
\[
\hat{\tau} \leftarrow \text{the difference between the left and right } \tau\text{-degrees of } B_1(t)
\]
\[
[\hat{B}_2(t)] \leftarrow \text{RECRATINT}(s_2, \hat{L}_{s_2}, \hat{R}_{s_2}, N_s/2, \hat{\tau})
\]
\[
B(t) \leftarrow B_1(t) \cdot \hat{B}_2(t)
\]
else
\[
[B(t)] \leftarrow \text{RATINT}(s, L_s, R_s, N_s, \tau)
\]
end if

Note that \text{RECRATINT} calls \text{RATINT} with the parameter \tau. This corresponds to an initialization that is different from the one used in the version that we have presented in Section 1. The difference is that instead of the classical degree the \tau-degree is used. We omit the details.

We now have a divide and conquer algorithm for rational interpolation. This algorithm can be stabilized in three ways: by giving “difficult points” an adequate treatment, via iterative refinement and via downdating.

### 3.1. Difficult points.

At the lowest level each interpolation problem consists of a set of interpolation conditions that are to be satisfied. Our fast algorithm \text{RATINT} constructs a solution iteratively by adding interpolation points one by one. Pivoting is used to enhance the numerical stability. Some interpolation points may have residuals with respect to the subproblem solution that are very small: the subproblem may be close to degenerate. If we process these interpolation points, then the accuracy is likely to decrease. These points are therefore marked as “difficult.” They are not added at the subproblem level. Instead, they are put aside and handled only at the very end, after \text{RECRATINT} has finished, via the fast-only algorithm \text{RATINT}. If at this stage the corresponding transformed residuals are still small, this indicates that the problem is ill-conditioned. The overall complexity of our algorithm will be \(O(n \log^2 n)\) as long as the number of difficult points is not too large.

### 3.2. Iterative improvement.

\text{RECRATINT} computes a basis matrix \(B(z)\) that corresponds to the \(N_s\) interpolation points \(s\) having initial left and right residual vectors \(L_s\) and \(R_s\). If there are no difficult points, then \(B(z)\) has the following degree-structure:

\[
\begin{bmatrix}
= N_s/2 & < N_s/2 \\
< N_s/2 & = N_s/2
\end{bmatrix}
\]

In general, however, \text{RECRATINT} will discover difficult points. If the number of difficult points is equal to \(n_{\text{bad}}\), then \(B(z)\) has the degree-structure

\[
\begin{bmatrix}
= \alpha & < \alpha \\
< \beta & = \beta
\end{bmatrix}
\]

with \(\alpha + \beta + n_{\text{bad}} = N_s\). Therefore \(B(z)\) is not only \(\tau\)-(column) reduced but also row reduced. Its row highest degree coefficient is equal to the identity matrix.
Let us introduce the following notations. The component polynomials of $B(z)$ are denoted as $a(z), b(z), c(z)$ and $d(z),$

$$B(z) = \begin{bmatrix} a(z) & c(z) \\ b(z) & d(z) \end{bmatrix}.$$  

Let $s^+$ be a complex column vector containing all the “easy” points, i.e., all the interpolation points except the difficult ones. Let $L_{s+}$ and $R_{s+}$ be the corresponding parts of $L_s$ and $R_s$. Note that these vectors have size $\alpha + \beta$. We denote their components by $s_i^+, L_{s+}, i$ and $R_{s+}, i$ respectively ($i = 1, \ldots, \alpha + \beta$).

Let $p$ be a complex column vector of size $\#p$ with components $p_i, i = 1, \ldots, \#p$. Let $L_p$ and $R_p$ be complex column vectors of size $\#p$. Let $k, l \in \mathbb{N}$. Then $V_k(L_p, p)$ is defined as the scaled Vandermonde matrix

$$V_k(L_p, p) := \text{diag}(L_p) \left[ \begin{array}{c} p_j^{i-1} j=0,1,\ldots, k \\ i=1,\ldots, \#p \end{array} \right].$$

Note that $V_k(L_p, p)$ is a matrix of size $\#p \times (k+1)$. The last column of $V_k(L_p, p)$ is denoted as $v_k(L_p, p)$. The coupled Vandermonde matrix $V_{k,l}(L_p, R_p, p)$ is defined as

$$V_{k,l}(L_p, R_p, p) := \left[ \begin{array}{c} V_k(L_p, p) \\ V_l(R_p, p) \end{array} \right].$$

It is a matrix of size $\#p \times (k+l+2)$.

Let $p(z) \in \mathbb{C}[z]$ be a polynomial and let $\delta \in \mathbb{N}$ be an arbitrary upper bound for its degree. Then the stacking vector $\hat{p}_k \in \mathbb{C}^{\delta + 1}$ of $p(z)$ with respect to $\delta$ is defined by the equation

$$\left[ 1 \quad z \quad \cdots \quad z^\delta \right] \hat{p}_k = p(z).$$

The vector $\hat{p}_k \in \mathbb{C}^{\delta}$ is obtained by deleting the last component of $\hat{p}_k$. Let $p_k$ be the coefficient of $z^i$ in $p(z)$ for $i = 0, 1, 2, \ldots$. Of course, $p_k = 0$ if $i > \deg p(z)$.

The stacking vectors $\hat{a}_\alpha, \hat{b}_\beta, \hat{c}_\alpha$ and $\hat{d}_\beta$ satisfy the following linear system of homogeneous equations:

$$V_{\alpha, \beta}(L_{s+}, R_{s+}, s^+) \left[ \begin{array}{c} \hat{a}_\alpha \\ \hat{b}_\beta \\ \hat{c}_\alpha \\ \hat{d}_\beta \end{array} \right] = \left[ \begin{array}{c} 0 \\ 0 \\ \vdots \\ 0 \end{array} \right]$$

or

$$V_{\alpha, \beta}(L_{s+}, R_{s+}, s^+) \left[ \begin{array}{c} \hat{a}_\alpha \\ \hat{b}_\beta \\ \hat{c}_\alpha \\ \hat{d}_\beta \end{array} \right] = \left[ \begin{array}{c} -v_\alpha(L_{s+}, s^+) \\ -v_\beta(R_{s+}, s^+) \end{array} \right].$$

Note that $V_{\alpha, \beta}(L_{s+}, R_{s+}, s^+)$ is a square matrix. To enhance the accuracy of the computed approximations for $\hat{a}_\alpha, \hat{b}_\beta, \hat{c}_\alpha$ and $\hat{d}_\beta$, we will use iterative refinement based on an inversion formula for this matrix.

The following theorem provides us with the parameters for an inversion formula for coupled Vandermonde matrices.

**Theorem 66.** Let $V_{\alpha, \beta}(L_{s+}, R_{s+}, s^+)$ be nonsingular and

$$B(z) = \begin{bmatrix} a(z) & c(z) \\ b(z) & d(z) \end{bmatrix}$$
as defined by (84). Then the $3 \times 2$ system of linear equations

$$
\begin{align*}
B(s_+^i)h_i &= \begin{bmatrix} a(s_+^i) & c(s_+^i) \\ b(s_+^i) & d(s_+^i) \end{bmatrix} \begin{bmatrix} h_{i,1} \\ h_{i,2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \\
L_{s^+,i} & R_{s^+,i} \end{bmatrix} B'(s_+^i)h_i = 1
\end{align*}
$$

has a unique solution $h_i = \begin{bmatrix} h_{i,1} \\ h_{i,2} \end{bmatrix}$ in $\mathbb{C}^{2 \times 1}$ for every $i \in \{1, \ldots, \alpha + \beta\}$.

**Proof.** See Heinig [136, Proposition 4.2].

**Theorem 67.** Let $V := V_{\alpha-1,\beta-1}(L_{s^+,R_{s^+}},s^+)$ be nonsingular and

$$
B(z) = \begin{bmatrix} a(z) & c(z) \\ b(z) & d(z) \end{bmatrix}
$$

as defined by (84). Then the inverse of $V$ is given by

$$
V^{-1} = W V_{\alpha-1,\beta-1}(H_1,H_2,s^+)^T
$$

with

$$
W := \begin{bmatrix} A & C \\ B & D \end{bmatrix} \in \mathbb{C}^{(\alpha+\beta) \times (\alpha+\beta)}
$$

where

$$
A := \begin{bmatrix} a_1 & \cdots & a_\alpha \\ \vdots & \ddots & \vdots \\ a_\alpha \end{bmatrix} \in \mathbb{C}^{\alpha \times \alpha}, \quad B := \begin{bmatrix} b_1 & \cdots & b_\alpha \\ \vdots & \ddots & \vdots \\ b_\beta \end{bmatrix} \in \mathbb{C}^{\beta \times \alpha},
$$

$$
C := \begin{bmatrix} c_1 & \cdots & c_\beta \\ \vdots & \ddots & \vdots \\ c_\alpha \end{bmatrix} \in \mathbb{C}^{\alpha \times \beta}, \quad D := \begin{bmatrix} d_1 & \cdots & d_\beta \\ \vdots & \ddots & \vdots \\ d_\beta \end{bmatrix} \in \mathbb{C}^{\beta \times \beta}
$$

are upper triangular Hankel matrices. The complex column vectors $H_1,H_2 \in \mathbb{C}^{\alpha+\beta}$ are defined as

$$
H_1 = \begin{bmatrix} h_{1,1} \\ \vdots \\ h_{\alpha+\beta,1} \end{bmatrix} \quad \text{and} \quad H_2 = \begin{bmatrix} h_{1,2} \\ \vdots \\ h_{\alpha+\beta,2} \end{bmatrix}
$$

where $h_{i,1}$ and $h_{i,2}, i = 1, \ldots, \alpha + \beta$, are as in the previous theorem.

**Proof.** See Heinig [136, Theorem 4.1].

In the setting that we will consider in the next chapter, the interpolation points $s^+$ together with the difficult points will form a set of $N_s$ points that are equally spaced on the unit circle. The computation of the inversion parameters $H_1$ and $H_2$ can therefore be done in $O(N_s \log N_s)$ flops. Each iterative refinement step based on the inversion formula of Theorem 67 also involves $O(N_s \log N_s)$ flops. Note that iterative refinement can be applied at one or more intermediate levels.
3.3. Downdating. Finite precision arithmetic can lead to a situation where
\[ f_k^T B(s_k) \neq \begin{bmatrix} 0 & 0 \end{bmatrix} \]
for one or more interpolation points \( s_k \). As the matrix \( B(s_k) \) is singular, there exists a vector \( v \in \mathbb{C}^2 \) such that
\[ B(s_k)v = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \]
Define
\[ B(t) = \begin{bmatrix} B_L(t) & B_R(t) \end{bmatrix}, \quad v = \begin{bmatrix} v_L \\ v_R \end{bmatrix} \]
and let \( \alpha_L := \text{deg}B_L(t) \) and \( \alpha_R := \text{deg}B_R(t) \). If \( \alpha_L \geq \alpha_R \) and \( v_L \neq 0 \), then we replace \( B_L(t) \) by
\[ B_L(t) \leftarrow B(t)v/(t - s_k). \]
If, on the other hand, \( \alpha_L < \alpha_R \) and \( v_R \neq 0 \), then we replace \( B_R(t) \) by
\[ B_R(t) \leftarrow B(t)v/(t - s_k). \]
If \( v_L = 0 \), then \( B_R(t) \) is divisible by \( t - s_k \). Similarly, if \( v_R = 0 \), then \( B_L(t) \) is divisible by \( t - s_k \). These considerations lead to the following algorithm.

**Function** \([B(t), s_{\text{bad}}] \leftarrow \text{DOWNDATING}(B(t), s, L_s, R_s, N_s)\)

- \( N_s \): the number of interpolation conditions
- \( s \in \mathbb{C}^{N_s \times 1} \): the (mutually distinct) interpolation points
- \( L_s, R_s \in \mathbb{C}^{N_s \times 1} \): the initial left and right residual vectors
- \( B(t) \in \mathbb{C}[t]^{2 \times 2} \)
- **on input**: a basis matrix corresponding to the given interpolation data
- **on output**: the corresponding downrated basis matrix
- \( s_{\text{bad}} \): a complex column vector containing the interpolation points that have been downrated

\( s_{\text{bad}} \leftarrow \emptyset \)

**for** \( k = 1(1)N_s \)

**if** \( \| [ L_s(k) \quad R_s(k) ] \| > \eta \) **then**

Choose \( v \in \mathbb{C}^2 \) such that \( B(s(k))v = \begin{bmatrix} 0 & 0 \end{bmatrix}^T \) and \( \| v \| = 1 \)

- **let** \( B(t) = \begin{bmatrix} B_L(t) & B_R(t) \end{bmatrix} \) and \( v = \begin{bmatrix} v_L \\ v_R \end{bmatrix}^T \)
- \( \alpha_L \leftarrow \text{deg}B_L(t) \)
- \( \alpha_R \leftarrow \text{deg}B_R(t) \)

**if** \( \alpha_L \geq \alpha_R \) **then**

**if** \( v_L \neq 0 \) **then**

\( B_L(t) \leftarrow B(t)v/(t - s(k)) \)

**else**

\( B_R(t) \leftarrow B_R(t)/(t - s(k)) \)

**end if**

**else**

**if** \( v_R \neq 0 \) **then**

\( B_R(t) \leftarrow B(t)v/(t - s(k)) \)

**else**

\( B_L(t) \leftarrow B_L(t)/(t - s(k)) \)

**end if**

**end if**
We are now ready to formulate the stabilized version of our algorithm.

**recursive function** \( B(t), s_{bad} \) \( \leftarrow \) **RecRatInt** \((s, L_s, R_s, N_s, \tau)\)

-- \( \tau \in \mathbb{Z} \)
-- \( N_s = 2^{p+1} \) for some \( p \in \mathbb{N} \): the number of interpolation conditions
-- \( s \in \mathbb{C}^{N_s \times 1} \): the (mutually distinct) interpolation points
-- \( L_s, R_s \in \mathbb{C}^{N_s \times 1} \): the initial left and right residual vectors
-- \( B(t) \in \mathbb{C}^{l^2 \times 2} \): a \( \tau \)-reduced basis matrix corresponding to the given interpolation data
-- \( s_{bad} \): a complex column vector containing the difficult interpolation points

if \( N_s > 2^{\text{limit}} \) then
  \[
  \begin{bmatrix}
  s_1, L_{s_1}, R_{s_1}, s_2, L_{s_2}, R_{s_2}
  \end{bmatrix} \leftarrow \text{Split}(s, L_s, R_s)
  \]
  \[
  B_1(t), s_{\text{bad},1} \leftarrow \text{RecRatInt}(s_1, L_{s_1}, R_{s_1}, N_s/2, \tau)
  \]
  for \( k = 1(1)N_s/2 \)
    \[
    \begin{bmatrix}
    \tilde{L}_{s_2}(k) \quad \tilde{R}_{s_2}(k)
    \end{bmatrix} \leftarrow [L_{s_2}(k) \quad R_{s_2}(k)] \cdot B_1(s_2(k))
    \]
  end for
  \[
  \hat{\tau} \leftarrow \text{the difference between the left and right } \tau\text{-degrees of } B_1(t)
  \]
  \[
  \hat{B}_2(t), s_{\text{bad},2} \leftarrow \text{RecRatInt}(s_2, \tilde{L}_{s_2}, \tilde{R}_{s_2}, N_s/2, \hat{\tau})
  \]
  \[
  B(t) \leftarrow B_1(t) \cdot \hat{B}_2(t)
  \]
  \[
  s_{\text{bad}} \leftarrow s_{\text{bad},1} \oplus s_{\text{bad},2}
  \]
else
  \[
  B(t), s_{\text{bad}} \leftarrow \text{RatInt}(s, L_s, R_s, N_s, \tau)
  \]
end if

if \( N_s = 2^{\text{down}} \) then
  \[
  s^+ \leftarrow s \oplus s_{\text{bad}}
  \]
  \[
  B(t), s_{\text{bad},3} \leftarrow \text{Downdating}(s^+, L_s^+, R_s^+, N_s)
  \]
  \[
  s_{\text{bad}} \leftarrow s_{\text{bad}} \oplus s_{\text{bad},3}
  \]
end if

if \( N_s = 2^{\text{ref}} \) then
  \[
  s^+ \leftarrow s \oplus s_{\text{bad}}
  \]
  \[
  B(t) \leftarrow \text{ItRef}(B(t), s^+, L_s^+, R_s^+, N_s, N_{\text{ref}})
  \]
end if

**function** \( B(t) \) \( \leftarrow \) **RatIntAll** \((s, L_s, R_s, N_s, \tau)\)

-- \( \tau \in \mathbb{Z} \)
-- \( N_s = 2^{p+1} \) for some \( p \in \mathbb{N} \): the number of interpolation conditions
-- \( s \in \mathbb{C}^{N_s \times 1} \): the (mutually distinct) interpolation points
-- \( L_s, R_s \in \mathbb{C}^{N_s \times 1} \): the initial left and right residual vectors
-- \( B(t) \in \mathbb{C}^{l^2 \times 2} \): a \( \tau \)-reduced basis matrix corresponding to the given interpolation data

\[
B^+(t), s_{\text{bad}} \leftarrow \text{RecRatInt}(s, L_s, R_s, N_s, \tau)
\]
$N_{bad} \leftarrow \text{size}(s_{bad})$

if $N_{bad} > 0$ then
  calculate $L_{bad}$ and $R_{bad}$
  $\tau^- \leftarrow$ the difference between the left and right $\tau$-degrees of $B^+(t)$
  $[B^-(t)] \leftarrow \text{RATINT}(s_{bad}, L_{bad}, R_{bad}, N_{bad}, \tau^-)$
  $B(t) \leftarrow B^+(t) \cdot B^-(t)$
end if

return

Acknowledgements

The results in this chapter were obtained in close collaboration with Marc Van Barel.
CHAPTER 9

Superfast Hankel and Toeplitz solvers

In a Toeplitz matrix the entries are the same along each diagonal whereas in a Hankel matrix the entries are the same along each antidiagonal. These matrices occur in many applications, for example in signal processing or Markov chains [34, 213]. They also play a central role in the theory of orthogonal polynomials and Padé approximation. We have already encountered Hankel matrices in the first part of this thesis. A Toeplitz system of linear equations can be transformed into a Hankel system of linear equations by reversing the order of the unknowns and in this sense Toeplitz and Hankel systems are equivalent. Gaussian elimination requires $O(n^3)$ floating point operations to solve an $n \times n$ Toeplitz or Hankel system. However, Toeplitz and Hankel matrices depend on $2n - 1$ instead of $n^2$ entries and hence one can try to exploit this structure to obtain algorithms that require less floating point operations than general purpose algorithms. An algorithm for solving Toeplitz or Hankel systems is called fast if its arithmetic complexity is $O(n^2)$. Superfast algorithms require only $O(n \log^2 n)$ flops.

Several fast algorithms exist. They compute a triangular factorization of the matrix or of its inverse in a recursive way. More precisely, in step $k \leq n$ a factorization of the $k \times k$ leading principal submatrix (section) is constructed based on a factorization of the $(k-1) \times (k-1)$ section. These algorithms only work if all these sections are nonsingular, i.e., if the matrix is strongly nonsingular. There exists algorithms that overcome this difficulty by jumping over exactly singular sections. These approaches result in a block triangular factorization of the matrix. Nearly singular sections are handled in the same way as nonsingular sections. Hence it is safe to apply these methods in exact arithmetic. However, in floating point arithmetic, stability problems are likely to occur in case nearly singular sections are involved. We refer the reader to [35, 170] and the references cited therein.

Look-ahead algorithms try to overcome this instability by looking ahead from one well-conditioned section to the next one and by jumping over the ill-conditioned sections that lie in between. If there are only a few such ill-conditioned sections, then the jumps can be small and the algorithm remains fast. However, if all the sections of the matrix except the last one (the matrix itself) are singular, then look-ahead algorithms will require $O(n^3)$ flops. Also, a look-ahead criterion and a condition estimator that are reliable as well as cheap are difficult to construct. One has to balance between a severe look-ahead criterion that can lead to big jumps and a less stringent criterion that can give less accurate results. For more details about look-ahead algorithms for Toeplitz systems we refer to [57, 58, 84, 85, 89, 113, 123, 129, 132, 133, 150]. Look-ahead algorithms for Hankel systems can be
found in [39, 51, 86]. For the block Toeplitz and the block Hankel case, the reader may consult [277]. In [88] a look-ahead Schur algorithm was designed for Hermitian block Toeplitz matrices. Several high performance algorithms for Toeplitz and block Toeplitz matrices are described in [90], including two look-ahead Schur algorithms for symmetric indefinite block Toeplitz matrices. In [250] a look-ahead block Schur algorithm for Toeplitz-like matrices was presented.

Good look-ahead strategies are difficult to design. Only recently a completely different approach has been considered. Gaussian elimination relies on (partial or complete) pivoting to enhance the numerical stability. Unfortunately, pivoting destroys the structure of a Hankel or a Toeplitz matrix. Other classes of matrices maintain their structure after pivoting. For example, a matrix $M = [m_{kl}]$ is called Cauchy-like if there exist certain numbers $y_k$ and $z_l$ such that the rank of the matrix $[(y_k - z_l)m_{kl}]$ is small compared to the order of $M$. Pivoting does not destroy the Cauchy-like structure and hence fast as well as numerically stable algorithms for solving Cauchy-like systems can be designed [104, 105, 106, 134, 140, 170]. Heinig [134] was one of the first who proposed to transform structured matrices from one class into another and to use pivoting strategies to enhance the numerical stability. For an overview of different transformation techniques and algorithms we refer the reader to [90, 137, 138, 139] and the references cited therein.

Pivoting for structured matrices has a good record of numerical performance in several applications. For example, what is known as the Leja ordering is the equivalent of partial pivoting for Vandermonde-related structures, see [143, 247]. Rational Leja ordering (i.e., for Cauchy matrices) is discussed in [40]. In the context of interpolation, pivoting is numerically shown to be a very successful technique in [109] and [169]. Numerical experiments in [107] and [169] showed the practicality of this approach for shift-invariant structures (such as Toeplitz or Toeplitz plus Hankel matrices). For a review about pivoting on structured matrices, we refer to [110].

In [135] Heinig transformed a Toeplitz system into a paired Vandermonde system, which is then solved as a tangential Lagrange interpolation problem. When two steps of his algorithm are combined, one obtains a block-step algorithm [137]. In this chapter we will present a fast Hankel solver, a fast block Hankel solver and a superfast Hankel solver. Our approach is related to Heinig's in the sense that they both proceed by first transforming the system and they both involve the solution of interpolation problems. We will also present a superfast Toeplitz solver that is based on a different transformation approach. In Section 1 we will show how an $n \times n$ Hankel system $Hx = b$ can be transformed into a Loewner system $Lx' = b'$. An explicit formula for $L^{-1}$ enables us to calculate $x'$ as $L^{-1}b'$. The solution $x'$ is then transformed back into $x$. These steps require only $O(n \log n)$ flops. The inversion formula for Loewner matrices involves certain parameters that can be computed by solving two linearized rational interpolation problems. By using the $O(n^2)$ algorithm RATINT that we have presented in the previous chapter, we obtain a fast Hankel solver. Note that RATINT incorporates pivoting. The corresponding algorithm for block Hankel matrices will be presented in Section 2.
The first superfast algorithms were designed by Sugiyama et al. [262], Bitmead and Anderson [37], Brent, Gustavson and Yun [43] and also Morf [214]. More recent algorithms can be found in [4, 5, 6, 49, 64, 112, 194, 221, 261]. For the block case we refer to [102, 196]. The main disadvantage of these algorithms is that they cannot handle nearly singular leading principal submatrices. To overcome this problem, Gutknecht [129] and Gutknecht and Hochbruck [132, 133] developed an algorithm that combines the look-ahead idea with divide and conquer techniques. Because in most practical problems the look-ahead step will be small compared to the order of the system that is to be solved, the algorithm is \textit{generically} superfast. In the previous chapter we have used a divide and conquer approach to obtain a superfast algorithm for rational interpolation. By using this algorithm instead of \textsc{RatInt} we obtain a stabilized generically superfast solver for indefinite Hankel systems, although only in case the order \( n \) is a power of 2. This will be discussed in Section 3. In Section 4 we will present a stabilized generically superfast solver for indefinite Toeplitz systems. Our approach relies on an inversion formula for Toeplitz matrices and is valid for arbitrary \( n \). This algorithm is faster than our superfast Hankel solver. The inversion formula contains certain parameters that are computed by solving linearized rational interpolation problems. Numerical examples will illustrate the effectiveness of our approach.

The algorithms that we will present in this chapter have been implemented in Fortran 90 as well as in Matlab. Our software is available at

\url{http://www.cs.kuleuven.ac.be/~marc/hankel/}

Part of this chapter is contained in our papers [187], [189], [278] and [279].

1. A fast Hankel solver

Let \( n \) be a positive integer, let \( H = H_n := [h_{k+l}]_{k,l=0}^{n-1} \) be a nonsingular \( n \times n \) complex Hankel matrix and let \( b \in \mathbb{C}^n \). We consider the problem of computing \( x := H^{-1}b \).

1.1. Transformation into a Loewner system. Let \( y_1, \ldots, y_n, z_1, \ldots, z_n \) be \( 2n \) mutually distinct complex numbers and define \( y := (y_1, \ldots, y_n) \) and \( z := (z_1, \ldots, z_n) \). Let \( L(y, z) \) be the class of matrices

\[
L(y, z) := \left\{ \left[ \frac{c_k - d_l}{y_k - z_l} \right]_{k,l=1}^{n} \mid c_1, \ldots, c_n, d_1, \ldots, d_n \in \mathbb{C} \right\}.
\]

The elements of \( L(y, z) \) are called \textit{Loewner matrices}. They bear the name of Karl Loewner who studied them in the context of rational interpolation and monotone matrix functions [202].

The set \( L(y, z) \) is a linear space over \( \mathbb{C} \) and a subspace of the linear space of all the \( n \times n \) complex matrices. Since addition of a constant to all the \( 2n \) parameters \( c_k, d_l \) leads to the same Loewner matrix, its dimension is \( 2n - 1 \). The set of all the \( n \times n \) complex Hankel matrices also forms a linear subspace of dimension \( 2n - 1 \). Hankel and Loewner matrices are even more closely related. According to Fiedler [79] every Hankel matrix can be transformed into a Loewner matrix and vice versa. We will now formulate this theorem and discuss our \( \mathcal{O}(n \log n) \) implementation of the transformation.

\[
\begin{align*}
\end{align*}
\]
First we have to deal with some preliminaries concerning Vandermonde matrices. Let \( t_1, \ldots, t_n \) be \( n \) complex numbers and define \( t := (t_1, \ldots, t_n) \). The Vandermonde matrix with nodes \( t_1, \ldots, t_n \) is given by

\[
V(t) = V(t_1, \ldots, t_n) := \begin{bmatrix}
1 & t_1 & \cdots & t_1^{n-1} \\
\vdots & \vdots & & \vdots \\
1 & t_n & \cdots & t_n^{n-1}
\end{bmatrix}.
\]

Let \( f_t(z) \) be the monic polynomial of degree \( n \) that has zeros \( t_1, \ldots, t_n \),

\[
f_t(z) := (z - t_1) \cdots (z - t_n),
\]

and define

\[
f_{k, j}(z) := \prod_{k \neq j} (z - t_k), \quad j = 1, \ldots, n.
\]

Note that \( f_{k, j}(z) \) is a monic polynomial of degree \( n - 1 \) for \( j = 1, \ldots, n \). Define the \( n \times n \) matrix \( W(t) \) by the equation

\[
\begin{bmatrix}
f_{t, 1}(z) \\
f_{t, 2}(z) \\
\vdots \\
f_{t, n}(z)
\end{bmatrix} = W(t) \begin{bmatrix} 1 \\ z \\ \vdots \\ z^{n-1} \end{bmatrix}.
\]

This means that the \( j \)th row of \( W(t) \) contains the coefficients of \( f_{k, j}(z) \) when written in terms of the standard monomial basis \( \{1, z, \ldots, z^{n-1}\} \). Then

\[
W(t) [V(t)^T = \text{diag}(f_{t, 1}(t_1), \ldots, f_{t, n}(t_n)) =: D(t).
\]

The Vandermonde matrix \( V(t) \) is nonsingular if and only if its nodes \( t_1, \ldots, t_n \) are mutually distinct. In that case \( (86) \) implies that \( W(t) \) is nonsingular and also that

\[
[V(t)]^{-1} = [W(t)^T [D(t)]^{-1}.
\]

For an early reference to this formula see, for example, the book by Kowalewski [181].

Let \( V(y, z) \) be the \( 2n \times 2n \) Vandermonde matrix with nodes \( y_1, \ldots, y_n \) and \( z_1, \ldots, z_n \) and similarly for \( W(y, z) \).

**Theorem 68.** The matrix \( L := W(y) H [W(z)^T \) is a Loewner matrix in \( \mathcal{L}(y, z) \) whose parameters \( c_1, \ldots, c_n, d_1, \ldots, d_n \) are given by (up to an arbitrary additive constant \( \xi \in \mathbb{C} \))

\[
\begin{bmatrix}
c_1 \\
\vdots \\
c_n \\
d_1 \\
\vdots \\
d_n
\end{bmatrix} = W(y, z) \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{2n-2} \\ \xi
\end{bmatrix}.
\]

**Proof.** See Fiedler [79, Theorem 12]. \( \square \)
Note that $L$ is nonsingular.

A judicious choice of the points $y$ and $z$ enables us to transform the Hankel system $Hx = b$ into the Loewner system $Lx' = b'$ in $O(n \log n)$ flops. Let $\omega := \exp(2\pi i/n)$ and suppose from now on that $y_k = \omega^{k-1}$ for $k = 1, \ldots, n$. That is, let $y = (1, \omega, \ldots, \omega^{n-1})$. Let $\zeta := \exp(\pi i/n)$ and suppose from now on that $z_k = \zeta y_k$ for $k = 1, \ldots, n$. That is, let $z = (\zeta, \zeta^2, \ldots, \zeta^{2n-1})$. Note that

$$y = (1, \zeta^2, \ldots, \zeta^{2n-2}) \quad \text{and} \quad z = (\zeta, \zeta^3, \ldots, \zeta^{2n-1}).$$

Let $\Omega_n$ be the $n \times n$ Fourier matrix,

$$(88) \quad \Omega_n := \frac{1}{\sqrt{n}} V(1, \omega, \ldots, \omega^{n-1}).$$

Matrix-vector products involving $\Omega_n$ ($\Omega_n^H$) amount to a (inverse) discrete Fourier transform (DFT) and can be evaluated via the celebrated (inverse) fast Fourier transform (FFT) in $O(n \log n)$ flops. Finally, let $D_{n,\omega}$ and $D_{n,\zeta}$ be the $n \times n$ diagonal matrices $D_{n,\omega} := \text{diag}(1, \omega, \ldots, \omega^{n-1})$ and $D_{n,\zeta} := \text{diag}(1, \zeta, \ldots, \zeta^{n-1})$.

**Theorem 69.** The solution to the Hankel system $Hx = b$ is given by

$$x = \sqrt{n} \zeta^{-1} D_{n,\zeta} \Omega_n^H D_{n,\omega} x'$$

where $x' := L^{-1} b'$ with $b' := \sqrt{n} D_{n,\omega} \Omega_n b$.

**Proof.** Theorem 68 implies that $x = H^{-1} b$ is given by $x = [W(z)]^T x'$ where $x' := L^{-1} b'$ with $b' := W(y) b$. Equation (86) and (88) and the fact that $\Omega_n$ is unitary imply that

$$(89) \quad W(y) = W(1, \omega, \ldots, \omega^{n-1}) = \frac{1}{\sqrt{n}} D(1, \omega, \ldots, \omega^{n-1}) \Omega_n.$$

An easy calculation shows that $D(1, \omega, \ldots, \omega^{n-1}) = n \overline{D_{n,\omega}}$ and thus $b' = W(y) b = \sqrt{n} D_{n,\omega} \Omega_n b$.

Equation (87) tells us that

$$[W(z)]^T = [V(z)]^{-1} D(z).$$

Since $V(z) = V(1, \omega, \ldots, \omega^{n-1}) D_{n,\zeta}$ and $D(z) = n \zeta^{-1} \overline{D_{n,\omega}}$, it follows that

$$x = [W(z)]^T x' = n \zeta^{-1} D_{n,\zeta}^{-1} D_{n,\omega}^{-1} [V(1, \omega, \ldots, \omega^{n-1})]^{-1} \overline{D_{n,\omega}} x' = \sqrt{n} \zeta^{-1} D_{n,\zeta} \Omega_n H \overline{D_{n,\omega}} x'.$$

This proves the theorem.

Let $P$ be the permutation matrix defined by

$$P \begin{bmatrix} y_1 \\ \vdots \\ y_n \\ z_1 \\ \vdots \\ z_n \end{bmatrix} = \begin{bmatrix} 1 \\ \zeta \\ \vdots \\ \zeta^{2n-1} \end{bmatrix}.$$
let \( \Omega_{2n} \) be the \( 2n \times 2n \) Fourier matrix,

\[
\Omega_{2n} := \frac{1}{\sqrt{2n}} V(1, \zeta, \ldots, \zeta^{2n-1}),
\]

and let \( D_{2n, \zeta} := \text{diag}(1, \zeta, \ldots, \zeta^{2n-1}) \).

**Theorem 70.** The parameters \( c_1, \ldots, c_n, d_1, \ldots, d_n \) of the Loewner matrix \( L \) are given by (up to an arbitrary additive constant \( \xi \in \mathbb{C} \))

\[
\begin{bmatrix}
  c_1 \\
  \vdots \\
  c_n \\
  d_1 \\
  \vdots \\
  d_n
\end{bmatrix} = \sqrt{2n} P^T D_{2n, \zeta} \Omega_{2n} \begin{bmatrix}
  h_0 \\
  h_1 \\
  \vdots \\
  h_{2n-2} \\
  \xi
\end{bmatrix}.
\]

**Proof.** Equation (85) and (89) imply that

\[
P W(y, z) = W(1, \zeta, \ldots, \zeta^{2n-1}) = \frac{1}{\sqrt{2n}} D(1, \zeta, \ldots, \zeta^{2n-1}) \Omega_{2n}.
\]

An easy calculation shows that \( D(1, \zeta, \ldots, \zeta^{2n-1}) = 2n D_{2n, \zeta} \) and thus

\[
W(y, z) = \sqrt{2n} P^T D_{2n, \zeta} \Omega_{2n}.
\]

The result then follows immediately from Theorem 68.

**Note.** In the proof of Theorem 69 we have shown that

\[
W(y) = \sqrt{n} D_n, \Omega_n \quad \text{and} \quad [W(z)]^T = \sqrt{n} \zeta^{n-1} D_n, \Omega_n H D_n, \omega.
\]

These formulae imply that \( W(y)/\sqrt{n} \) and \([W(z)]^T/\sqrt{n}\) are unitary. Therefore \( \|L\|_2 = n\|H\|_2 \) and \( \|L^{-1}\|_2 = n^{-1}\|H^{-1}\|_2 \) and thus \( \kappa_2(L) = \kappa_2(H) \). In other words, the spectral condition number is left unchanged.

### 1.2. An inversion formula for Loewner matrices.

**Theorem 71.** Let \( p_k, u_k, \tilde{p}_k \) and \( \tilde{u}_k \) for \( k = 1, \ldots, n \) be defined by the equations

\[
\begin{bmatrix}
  p_1 & \cdots & p_n \\
  u_1 & \cdots & u_n
\end{bmatrix} L = \begin{bmatrix}
  1 & \cdots & 1 \\
  d_1 & \cdots & d_n
\end{bmatrix}
\]

\[
L \begin{bmatrix}
  \tilde{p}_1 \\
  \vdots \\
  \tilde{p}_n
\end{bmatrix} = \begin{bmatrix}
  1 \\
  \vdots \\
  1
\end{bmatrix}
\]

\[
L \begin{bmatrix}
  \tilde{u}_1 \\
  \vdots \\
  \tilde{u}_n
\end{bmatrix} = \begin{bmatrix}
  c_1 \\
  \vdots \\
  c_n
\end{bmatrix}.
\]

Then the inverse of \( L \) is given by

\[
L^{-1} = \left[ \frac{\tilde{u}_k p - \tilde{p}_k u_k}{y - \frac{1}{\zeta}} \right]_{k,l=1}^{n}.
\]
A Loewner matrix is a special kind of Cauchy-like matrix. The inversion formula given in the previous theorem can therefore be seen as a special case of the formula that appears in the book written by Heinig and Rost \cite{heinig2010}. The parameters that appear in the inversion formula \eqref{eq:inversion_formula} can be computed by solving the following two linearized rational interpolation problems.

\textbf{Theorem 7.2.} Define
\[
\begin{align*}
  u(z) & := f_y(z) - \sum_{k=1}^{n} u_k f_{x,k}(z) \\
  v(z) & := -\sum_{k=1}^{n} c_k u_k f_{x,k}(z) \\
  \tilde{u}(z) & := f_a(z) - \sum_{k=1}^{n} \tilde{u}_k f_{x,k}(z) \\
  \tilde{v}(z) & := -\sum_{k=1}^{n} d_k \tilde{u}_k f_{x,k}(z).
\end{align*}
\]

Then the polynomial pair \((v(z), u(z))\) is the only pair such that
\begin{itemize}
  \item \(u(z) \in \mathbb{C}[z]\) and \(\deg u(z) = n\), \(v(z) \in \mathbb{C}[z]\) and \(\deg v(z) < n\), \(u(z)\) is monic
  \item \(v(y_k) = c_k u(y_k)\) and \(v(z_k) = d_k u(z_k)\) for \(k = 1, \ldots, n\).
\end{itemize}
The polynomial pair \((\tilde{v}(z), \tilde{u}(z))\) satisfies exactly the same properties and thus \(v(z) \equiv \tilde{v}(z)\) and \(u(z) \equiv \tilde{u}(z)\).

\textbf{Proof.} See Van Barel and Vavrín \cite{vandenberghe2013}, Theorem 3.1. Compare also with Vavrín \cite{vavrin2013}, Theorem 2.1. \(\Box\)

\textbf{Theorem 7.3.} Define
\[
\begin{align*}
  p(z) & := \sum_{k=1}^{n} p_k f_{x,k}(z) \\
  q(z) & := f_y(z) + \sum_{k=1}^{n} c_k p_k f_{y,k}(z) \\
  \tilde{p}(z) & := \sum_{k=1}^{n} \tilde{p}_k f_{x,k}(z) \\
  \tilde{q}(z) & := f_a(z) + \sum_{k=1}^{n} d_k \tilde{p}_k f_{x,k}(z).
\end{align*}
\]

Then the polynomial pair \((q(z), p(z))\) is the only pair such that
\begin{itemize}
  \item \(q(z) \in \mathbb{C}[z]\) and \(\deg q(z) = n\), \(p(z) \in \mathbb{C}[z]\) and \(\deg p(z) < n\), \(q(z)\) is monic
  \item \(q(y_k) = c_k p(y_k)\) and \(q(z_k) = d_k p(z_k)\) for \(k = 1, \ldots, n\).
\end{itemize}
The polynomial pair \((\tilde{q}(z), \tilde{p}(z))\) satisfies exactly the same properties and thus \(q(z) \equiv \tilde{q}(z)\) and \(p(z) \equiv \tilde{p}(z)\).
Note that if \( p(y_k) \) and \( p(z_k) \), \( k = 1, \ldots, n \), are different from zero, then the rational function \( q(z)/p(z) \) satisfies the proper rational interpolation conditions
\[
\frac{q(y_k)}{p(y_k)} = c_k \quad \text{and} \quad \frac{q(z_k)}{p(z_k)} = d_k, \quad k = 1, \ldots, n.
\]
Similarly, if \( u(y_k) \) and \( u(z_k) \), \( k = 1, \ldots, n \), are different from zero, then the rational function \( v(z)/u(z) \) satisfies the proper rational interpolation conditions
\[
\frac{v(y_k)}{u(y_k)} = c_k \quad \text{and} \quad \frac{v(z_k)}{u(z_k)} = d_k, \quad k = 1, \ldots, n.
\]
The rational functions \( q(z)/p(z) \) and \( v(z)/u(z) \) are different because their degree structure is different.

An easy calculation reveals the following connections with the parameters that appear in the inversion formula (90):
\[
\begin{bmatrix}
  u_1 \\
  \vdots \\
  u_n
\end{bmatrix} = -\frac{1}{n} D_{n,\omega} \begin{bmatrix}
  u(y_1) \\
  \vdots \\
  u(y_n)
\end{bmatrix}, \quad \begin{bmatrix}
  p_1 \\
  \vdots \\
  p_n
\end{bmatrix} = \frac{1}{n} D_{n,\omega} \begin{bmatrix}
  p(y_1) \\
  \vdots \\
  p(y_n)
\end{bmatrix},
\]
\[
\begin{bmatrix}
  \tilde{u}_1 \\
  \vdots \\
  \tilde{u}_n
\end{bmatrix} = -\frac{1}{n^{\frac{n-1}{2}}} D_{n,\omega} \begin{bmatrix}
  u(z_1) \\
  \vdots \\
  u(z_n)
\end{bmatrix}, \quad \begin{bmatrix}
  \tilde{p}_1 \\
  \vdots \\
  \tilde{p}_n
\end{bmatrix} = \frac{1}{n^{\frac{n-1}{2}}} D_{n,\omega} \begin{bmatrix}
  p(z_1) \\
  \vdots \\
  p(z_n)
\end{bmatrix}.
\]
As the points \( y_1, \ldots, y_n, z_1, \ldots, z_n \) are (up to a permutation) equal to the 2nth roots of unity \( 1, \zeta, \ldots, \zeta^{2n-1} \) we can use the FFT to evaluate the polynomials \( u(z) \) and \( p(z) \) at these points.

Once these inversion parameters have been computed, the matrix-vector product \( x' = L^{-1}y' \) can be calculated in \( O(n \log n) \) flops. Indeed, \( L^{-1} \) can be written as
\[
L^{-1} = \text{diag}(\tilde{p}_1, \ldots, \tilde{p}_n) C \text{diag}(u_1, \ldots, u_n) - \text{diag}(\tilde{u}_1, \ldots, \tilde{u}_n) C \text{diag}(p_1, \ldots, p_n)
\]
if \( C \) is given by the Cauchy matrix
\[
C := \left[ \frac{1}{z_k - y_l} \right]_{k,l=1}^n.
\]
By applying Proposition 3.2 in [80] to the special case of the roots of unity, one obtains that the Cauchy matrix \( C \) can be factorized as
\[
C = -\frac{n}{2} \Omega_n D_{n,\zeta} \Omega_n^H D_{n,\omega}.
\]
This implies that the product of \( C \) with a vector in \( \mathbb{C}^n \) can be evaluated in \( O(n \log n) \) flops and thus, because of (91), the same holds for the product \( L^{-1}y' \).
1.3. The algorithm RATINT. Let \( s_k := y_k \) and \( s_{n+k} := z_k \) for \( k = 1, \ldots, n \). Define the column vectors \( f_1, \ldots, f_{2n} \in \mathbb{C}^{2 \times 1} \) as
\[
f_k := \begin{bmatrix} 1 \\ -c_k \end{bmatrix}, \quad f_{n+k} := \begin{bmatrix} 1 \\ -d_k \end{bmatrix}, \quad k = 1, \ldots, n.
\]
The linearized rational interpolation problems that appear in Theorems 72 and 73 can be formulated as
\[
\begin{align*}
f_k^T B(s_k) &= [0 \ 0], \quad k = 1, \ldots, 2n,
\end{align*}
\]
where
\[
B(z) = \begin{bmatrix} n_\ell(z) & n_r(z) \\ d_\ell(z) & d_r(z) \end{bmatrix} \in \mathbb{C}[z]^{2 \times 2}
\]
with \( \deg n_\ell(z) = n \), \( \deg d_\ell(z) \leq n - 1 \), \( \deg n_r(z) \leq n - 1 \) and \( \deg d_r(z) = n \), and \( n_\ell(z) \) as well as \( d_r(z) \) monic, i.e., where \( B(z) \) is a monic \( 2 \times 2 \) matrix polynomial of degree \( n \). This corresponds to the interpolation problem that we have considered in Section 1 of Chapter 8. Theorems 72 and 73 assert that this problem has a unique solution given by
\[
B^*(z) := \begin{bmatrix} q(z) & v(z) \\ p(z) & u(z) \end{bmatrix}.
\]
Algorithm RATINT calculates a \( 2 \times 2 \) matrix polynomial \( B(z) \) of degree \( n \) that satisfies (93) and whose highest degree coefficient \( A \in \mathbb{C}^{2 \times 2} \) is nonsingular. This implies that \( B(z) \equiv B^*(z)A \). In Chapter 8 we have shown that \( \det A = 1 \).

Note that only the second row of \( B^*(z) \) is needed to compute the inversion parameters. If we compute the inversion parameters from the polynomials that appear in the second row of \( B(z) \), then we still obtain the correct value of \( L^{-1} \) because \( \det A = 1 \). Indeed, suppose that
\[
A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}
\]
and let \( p'(z) := ap(z) + cu(z) \) and \( u'(z) := bp(z) + du(z) \). Define
\[
\begin{align*}
\begin{bmatrix}
u'_1 \\ \vdots \\ u'_n
\end{bmatrix} &= -\frac{1}{n} D_{n,\omega} \begin{bmatrix}
u'(y_1) \\ \vdots \\ 
u'(y_n)
\end{bmatrix}, \\
\begin{bmatrix}
\hat{p}'_1 \\ \vdots \\ \hat{p}'_n
\end{bmatrix} &= \frac{1}{n} D_{n,\omega} \begin{bmatrix}
\hat{p}'(y_1) \\ \vdots \\ \hat{p}'(y_n)
\end{bmatrix}, \\
\begin{bmatrix}
\hat{\nu}'_1 \\ \vdots \\ \hat{\nu}'_n
\end{bmatrix} &= -\frac{1}{n} \frac{1}{\omega^{n-1}} D_{n,\omega} \begin{bmatrix}
u'(z_1) \\ \vdots \\ 
u'(z_n)
\end{bmatrix}, \\
\begin{bmatrix}
\hat{p}'_1 \\ \vdots \\ \hat{p}'_n
\end{bmatrix} &= \frac{1}{n} \frac{1}{\omega^{n-1}} D_{n,\omega} \begin{bmatrix}
\hat{p}'(z_1) \\ \vdots \\ \hat{p}'(z_n)
\end{bmatrix}.
\end{align*}
\]
Then
\[
\begin{align*}
u'_k &= -bp_k + du_k, \\
\hat{\nu}'_k &= -b\hat{p}_k + d\hat{u}_k,
\end{align*}
\]
for \( k = 1, \ldots, n \). This implies that
\[
\begin{bmatrix}
u'_k \\ \hat{\nu}'_k
\end{bmatrix} = \begin{bmatrix}
u_k \\ \hat{\nu}_k
\end{bmatrix}, \quad \begin{bmatrix}p_k \\ \hat{p}_k
\end{bmatrix} = \begin{bmatrix}p'_k \\ \hat{p}'_k
\end{bmatrix},
\]
for \( k = 1, \ldots, n \). This implies that
\[
\begin{bmatrix}
u'_k \\ \hat{\nu}'_k
\end{bmatrix} = \begin{bmatrix}
u_k \\ \hat{\nu}_k
\end{bmatrix}, \quad \begin{bmatrix}p_k \\ \hat{p}_k
\end{bmatrix} = \begin{bmatrix}p'_k \\ \hat{p}'_k
\end{bmatrix},
\]
for \( k = 1, \ldots, n \). This implies that
\[
\begin{align*}
\left[ \frac{\nu'_k}{\hat{\nu}'_k} \right]_{k=1}^n &= \left[ \frac{\nu_k}{\hat{\nu}_k} \right]_{k=1}^n, \\
\left[ \frac{p_k}{\hat{p}_k} \right]_{k=1}^n &= \left[ \frac{p'_k}{\hat{p}'_k} \right]_{k=1}^n.
\end{align*}
\]
because
\[ \tilde{d}_k p_k - \tilde{p}_k u_l = \det \begin{bmatrix} \tilde{a}_k & u_l \\ \tilde{p}_k & \tilde{p}_l \end{bmatrix} = \det \left( \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \begin{bmatrix} \tilde{a}_k & u_l \\ \tilde{p}_k & \tilde{p}_l \end{bmatrix} \right) = \det \begin{bmatrix} \tilde{a}_k & u_l \\ \tilde{p}_k & \tilde{p}_l \end{bmatrix} = \tilde{a}_k p_k - \tilde{p}_k u_l \]
for \( k, l = 1, \ldots, n \) since
\[
\begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} \det A & 0 \\ 0 & \det A \end{bmatrix} = I_2.
\]

What is the arithmetic complexity of this approach? It takes \( 2n^2 + \mathcal{O}(n) \) complex multiplications and \( 2n^2 + \mathcal{O}(n) \) complex additions to compute the second row of \( B(z) \). The updating of the residuals costs \( 4n^2 + \mathcal{O}(n) \) complex multiplications and \( 4n^2 + \mathcal{O}(n) \) complex additions. Thus the overall cost is \( 6n^2 + \mathcal{O}(n) \) complex multiplications and \( 6n^2 + \mathcal{O}(n) \) complex additions. Therefore, by solving the interpolation problems via \texttt{RatInt}, we obtain a fast Hankel solver, i.e., a Hankel solver that has arithmetic complexity \( \mathcal{O}(n^2) \). Indeed, it takes only \( \mathcal{O}(n \log n) \) flops to transform the Hankel system \( Hx = b \) into the Loewner system \( Lx' = b' \) and once the solutions to the two linearized rational interpolation problems are known, the parameters that appear in the inversion formula for \( L \) can be computed in \( \mathcal{O}(n \log n) \) flops. A Fortran 90 as well as a Matlab implementation are available. In the Fortran version the FFTs are calculated via \texttt{FFTPACK}. In the next subsection we will discuss a few numerical experiments. They show the effectiveness of our approach.

1.4. Numerical examples. As a matrix-vector product involving a Hankel matrix amounts to a convolution of two vectors or, equivalently, the product of two polynomials, the residue \( r := b - H\hat{x} \) can be calculated via FFT in \( \mathcal{O}(n \log n) \) flops [35]. This implies that improving an approximation for \( x \) iteratively does not add to the \( \mathcal{O}(n^2) \) complexity of our fast Hankel solver.

In the following examples we used the Fortran 90 version of our package. The calculations were done by an IBM SP2 with machine epsilon \( \epsilon \approx 0.1210^{-6} \) in single precision and \( \epsilon \approx 0.2210^{-15} \) in double precision.

Example 28. We consider single precision \( n \times n \) real Hankel matrices \( H_n \) whose entries are random uniformly distributed in \([-1,1]\) for \( n = 10 \) (10) 500. The right-hand sides \( b_n \in \mathbb{R}^n \) are calculated in double precision such that \( x_n := H_n^{-1} b_n = [1 \cdots 1]^T \). Figure 1 shows the results obtained by our algorithm (before and after one step of iterative improvement in which the residue was calculated in double precision) and the results obtained via Gaussian elimination with partial pivoting (GEPP) using the LAPACK routines \texttt{CGETRF} and \texttt{CGETRS}. We have plotted \( -\log_{10}(\|x_n - x_n\|_\infty / \|x_n\|_\infty) \). Also shown is \( \log_{10} \kappa(H_n) \). The condition number was calculated via LAPACKs routine \texttt{CGECON}. Note that after one step of iterative improvement the relative accuracy of the solution is of the order of the machine precision except for very ill-conditioned Hankel matrices.
Figure 1. \(-\log_{10} \left( \frac{\|\hat{x}_n - x_n\|_\infty}{\|x_n\|_\infty} \right)\) and \(\log_{10} \kappa_\infty(H_n)\) versus \(n\).
**Example 29.** In this test we consider the Hankel matrix

\[
H := \begin{bmatrix}
\left(\frac{1}{2}\right)^{n-1} & \cdots & \frac{1}{2} \\
\left(\frac{1}{2}\right)^{n-2} & \cdots & \frac{1}{2} \\
\vdots & \ddots & \vdots \\
\frac{1}{2} & \cdots & \left(\frac{1}{2}\right)^{n-2} \\
\epsilon & \frac{1}{2} & \cdots & \left(\frac{1}{2}\right)^{n-1}
\end{bmatrix} \in \mathbb{R}^{n \times n}.
\]

The corresponding Toeplitz matrix belongs to the class of the Kac-Murdock-Szego matrices (with \(\rho = \frac{1}{2}\)) [167]. For \(n = 3k + 1, k = 1, 2, \ldots\), this matrix is singular if \(\epsilon = 0\) and, consequently, ill-conditioned if \(\epsilon\) is set to a small number. We take \(n = 1000\) and let \(\epsilon = 10^{-q}\) for \(q = 0, 1, \ldots, 15\). We define the right-hand side \(b := [b_1 \cdots b_n]^T \in \mathbb{R}^n\) as

\[b_k := 2 + \epsilon - \frac{1}{2}k^{k-1} - \frac{1}{2}n-k, \quad k = 1, \ldots, n,
\]

because then \(x := H^{-1}b\) is equal to \([1 \cdots 1]^T\) as one can easily verify. Figure 2 shows the results obtained by our algorithm (with up to three steps of iterative improvement). The calculations were done in double precision. We have plotted \(-\log_{10}(\|\hat{x} - x\|_\infty / \|x\|_\infty)\) versus \(q = -\log_{10}\epsilon\). Also shown is \(\log_{10} \kappa_\infty(H)\). In Figure 3 we have plotted \(-\log_{10}(\|b - H \hat{x}\|_\infty / \|b\|_\infty)\). Note that in case the condition number is not too large, the relative error for the residue after one (or more) step(s) of iterative improvement is of the order of the machine precision.

**Example 30.** We consider the Hankel matrix

\[
H_n := \begin{bmatrix}
1 & 2 & \cdots & n \\
2 & \ddots & \vdots & \vdots \\
\vdots & \ddots & \ddots & \vdots \\
n & 0 & \cdots & 0
\end{bmatrix}
\]

for \(n = 1000\) \((100)10000\). One can easily verify that \(\|H_n^{-1}\|_\infty = \mathcal{O}(1/n)\). As \(\|H_n\|_\infty = n(n+1)/2\), this implies that \(\kappa_\infty(H_n) = \mathcal{O}(n)\). We define the right-hand side \(b_n = [b_1^{(n)} \cdots b_n^{(n)}]^T\) as

\[b_k^{(n)} := \frac{1}{2}n(n+1) - \frac{1}{2}(k-1)k, \quad k = 1, \ldots, n.
\]

Then the solution to \(H_n x_n = b_n\) is given by \(x_n = [1 \cdots 1]^T\) as one can easily verify. Figure 4 shows the results obtained by our algorithm (with up to two steps of iterative improvement). The calculations were done in double precision.

### 2. A fast block Hankel solver

We will now show how the results presented in the previous section can be generalized to the block case. This will lead to a fast solver for block Hankel systems.

Let \(n\) and \(p\) be positive integers. Consider a sequence \(H_0, \ldots, H_{2n-2}\) in \(\mathbb{C}^{p \times p}\) such that the \(np \times np\) block Hankel matrix \(H := [H_{k,l}]_{k,l=0}^{n-1}\) is nonsingular. Let \(b \in \mathbb{C}^{np}\). We consider the problem of computing \(x := H^{-1}b\).
2.1. Transformation into a block Loewner system. Let $y_1, \ldots, y_n$ and $z_1, \ldots, z_n$ be $2n$ mutually distinct complex numbers and define $y := (y_1, \ldots, y_n)$ and $z := (z_1, \ldots, z_n)$. Let $\mathcal{L}_p(y, z)$ be the class of $n \times n$ block matrices

$$\mathcal{L}_p(y, z) := \left\{ \begin{bmatrix} C_k - D_k & \gamma_k - \gamma_l \\ y_k - z_l & \delta_k - \delta_l \end{bmatrix} \mid k, l = 1, \ldots, n, C_1, \ldots, C_n, D_1, \ldots, D_n \in \mathbb{C}^{p \times p} \right\}.$$ 

The elements of $\mathcal{L}_p(y, z)$ are called block Loewner matrices.
Figure 3. $-\log_{10}\frac{\|b - H\hat{x}\|_\infty}{\|b\|_\infty}$ versus $-\log_{10} \epsilon$

The set $\mathcal{L}_p(y,z)$ is a $\mathbb{C}^{p \times p}$-module and a submodule of the $\mathbb{C}^{p \times p}$-module of all the complex $n_p \times n_p$ block matrices having $p \times p$ blocks. Since addition of a constant matrix to all the $2n$ $p \times p$ matrices $C_1, D_1$ leads to the same block Loewner matrix, its dimension is $2n - 1$. The set of all the complex $n_p \times n_p$ block Hankel matrices having $p \times p$ blocks also forms a submodule of dimension $2n - 1$. Block Hankel and block Loewner matrices are even more closely related. We have already seen how in the scalar case ($p = 1$) every Hankel matrix can be transformed into a Loewner matrix and vice versa, cf. Theorem 68. We will now formulate this result for the block case and discuss our $O(p^2 n \log n)$ implementation of the transformation.

Recall that the Kronecker product of two matrices $A = [a_{kl}]_{k,l=1}^{\alpha} \in \mathbb{C}^{\alpha \times \alpha}$ and $B \in \mathbb{C}^{\beta \times \beta}$ is defined as the block matrix

$$A \otimes B := [a_{kl}B]_{k,l=1}^{\alpha \times \beta} \in \mathbb{C}^{\alpha \beta \times \alpha \beta}.$$  

Let the matrices $W(y)$ and $W(z)$ be defined as in Subsection 1.1.

**Theorem 74.** The matrix $L := [W(y) \otimes I_p] H [W(z) \otimes I_p]^T$ is a block Loewner matrix in $\mathcal{L}_p(y,z)$ whose $p \times p$ matrices $C_1, \ldots, C_n, D_1, \ldots, D_n$ are given by (up to an arbitrary additive matrix $\xi \in \mathbb{C}^{p \times p}$)

$$
\begin{bmatrix}
C_1 \\
\vdots \\
C_n \\
D_1 \\
\vdots \\
D_n
\end{bmatrix} = [W(y,z) \otimes I_p] 
\begin{bmatrix}
H_0 \\
H_1 \\
\vdots \\
H_{2n-2} \\
\xi
\end{bmatrix}.
$$
Proof. The scalar case ($p = 1$) was proven by Fiedler [79], cf. Theorem 68. The generalization to the block case is straightforward.

The Kronecker product $A \otimes B$ is nonsingular if $A$ and $B$ are both nonsingular. This implies that $W(y) \otimes I_p$ and $W(z) \otimes I_p$ are nonsingular and thus $L$ is nonsingular.

A judicious choice of the points $y$ and $z$ enables us to transform the block Hankel system $Hx = b$ into a block Loewner system $Lx' = b'$ in $\mathcal{O}(p^2n \log n)$ flops.
Let $\omega := \exp(2\pi i/n)$ and suppose from now on that $y_k = \omega^{k-1}$ for $k = 1, \ldots, n$. Also, let $\zeta := \exp(\pi i/n)$ and suppose from now on that $z_k = \zeta y_k$ for $k = 1, \ldots, n$. Let the matrices $\Omega_n$, $D_{n,\omega}$ and $D_{n,\zeta}$ be defined as in Subsection 1.1.

**Theorem 75.** The solution to the block Hankel system $Hx = b$ is given by

$$x = \left[ \sqrt{n} \zeta^{n-1} D_{n,\zeta} \Omega_n^H D_{n,\omega} \right] x'$$

where $x' := L^{-1} b'$ with $b' := \left[ \sqrt{n} D_{n,\omega} \Omega_n \right] b$.

**Proof.** Theorem 74 implies that $x = H^{-1} b$ is given by $x = [W(z) \otimes I_p]^T x'$ where $x' := L^{-1} b'$ with $b' := [W(y) \otimes I_p] b$. In the proof of Theorem 69 it is shown that $W(y) = \sqrt{n} D_{n,\omega} \Omega_n$ and $[W(y)]^T = \sqrt{n} \zeta^{n-1} D_{n,\zeta} \Omega_n^H D_{n,\omega}$. As $[W(z)]^T = [W(z)]^T \otimes I_p$, the result follows. \qed

It follows that

$$\begin{bmatrix} x_r \\ x_{p+r} \\ \vdots \\ x_{(n-1)p+r} \end{bmatrix} = \sqrt{n} \zeta^{n-1} D_{n,\zeta} \Omega_n^H D_{n,\omega} \begin{bmatrix} x'_r \\ x'_{p+r} \\ \vdots \\ x'_{(n-1)p+r} \end{bmatrix}$$

and

$$\begin{bmatrix} b_r \\ b'_{p+r} \\ \vdots \\ b'_{(n-1)p+r} \end{bmatrix} = \sqrt{n} D_{n,\omega} \Omega_n \begin{bmatrix} b_r \\ b_{p+r} \\ \vdots \\ b_{(n-1)p+r} \end{bmatrix}$$

for $r = 1, \ldots, p$. Thus the transformations $b \mapsto b'$ and $x' \mapsto x$ can be done in $O(m \log n)$ floating point operations.

Let the matrices $P$, $\Omega_{2n}$ and $D_{2n,\zeta}$ be defined as in Subsection 1.1.

**Theorem 76.** The $p \times p$ matrices $C_1, \ldots, C_n, D_1, \ldots, D_n$ of the block Loewner matrix $L$ are given by (up to an arbitrary additive matrix $\xi \in \mathbb{C}^{p \times p}$)

$$\begin{bmatrix} C_1 \\ \vdots \\ C_n \\ D_1 \\ \vdots \\ D_n \end{bmatrix} = \left[ \sqrt{2n} P^T D_{2n,\zeta} \Omega_{2n} \right] \begin{bmatrix} H_0 \\ H_1 \\ \vdots \\ H_{2n-2} \\ \xi \end{bmatrix}$$

**Proof.** In Theorem 70 it is shown that $W(y, z) = \sqrt{2n} P^T D_{2n,\zeta} \Omega_{2n}$. The result then follows immediately from Theorem 74. \qed
The previous theorem implies that

\[
\begin{bmatrix}
(C_1)_{k,l} \\
\vdots \\
(C_n)_{k,l} \\
(D_1)_{k,l} \\
\vdots \\
(D_n)_{k,l}
\end{bmatrix} = \sqrt{2n} \ P^T \ D_{2n,\xi} \ \frac{1}{\sqrt{2n}} \begin{bmatrix}
(H_0)_{k,l} \\
(H_1)_{k,l} \\
\vdots \\
(H_{2n-2})_{k,l} \\
(H_n)_{k,l}
\end{bmatrix}
\]

for \( k, l = 1, \ldots, p \). Thus \( H \) can be transformed into \( L \) in \( O(p^2 n \log n) \) flops.

Note. As we have already mentioned in Subsection 1.1, the matrices \( W(y) / \sqrt{m} \) and \( [W(z)]^T / \sqrt{m} \) are unitary. The Kronecker product of two unitary matrices is a unitary matrix. As

\[
L = n \left[ \frac{1}{\sqrt{n}} W(y) \otimes I_p \right] H \left[ \frac{1}{\sqrt{n}} [W(z)]^T \otimes I_p \right],
\]

it follows that \( \|L\|_2 = n \|H\|_2 \) and \( \|L^{-1}\|_2 = n^{-1} \|H^{-1}\|_2 \) and thus \( \kappa_2(L) = \kappa_2(H) \). In other words, the spectral condition number is left unchanged.

2.2. An inversion formula for block Loewner matrices.

Theorem 77. Let \( P_k, U_k, \tilde{P}_k \) and \( \tilde{U}_k \in \mathbb{C}^{p \times p} \) for \( k = 1, \ldots, n \) be defined by the equations

\[
\begin{bmatrix}
P_1 & \cdots & P_n
\end{bmatrix} L = \begin{bmatrix}
I_p & \cdots & I_p
\end{bmatrix}
\]

\[
\begin{bmatrix}
U_1 & \cdots & U_n
\end{bmatrix} L = \begin{bmatrix}
D_1 & \cdots & D_n
\end{bmatrix}
\]

\[
\begin{bmatrix}
\tilde{P}_1 \\
\vdots \\
\tilde{P}_n
\end{bmatrix} = \begin{bmatrix}
I_p \\
\vdots \\
I_p
\end{bmatrix}
\]

\[
\begin{bmatrix}
\tilde{U}_1 \\
\vdots \\
\tilde{U}_n
\end{bmatrix} = \begin{bmatrix}
C_1 \\
\vdots \\
C_n
\end{bmatrix}
\]

Then the inverse of \( L \) is given by

\[
L^{-1} = \left[ \frac{\tilde{U}_k P_k - \tilde{P}_k U_k}{y_k - z_k} \right]_{k,l=1}^n.
\]

Proof. See Van Barel and Vavřín [280, Theorem 2.2].

The \( p \times p \) matrices that appear in the inversion formula (94) can be computed by solving the following linearized rational interpolation problems.
Theorem 78. Define

\[ U(z) := f_\gamma(z)I_p - \sum_{k=1}^n f_{\gamma,k}(z)U_k \]

\[ V(z) := -\sum_{k=1}^n f_{\gamma,k}(z)U_kC_k \]

\[ \hat{U}(z) := f_\alpha(z)I_p - \sum_{k=1}^n f_{\alpha,k}(z)\hat{U}_k \]

\[ \hat{V}(z) := -\sum_{k=1}^n f_{\alpha,k}(z)D_k\hat{U}_k. \]

Then the pair \((V(z), U(z))\) is the only pair such that

- \(U(z) \in \mathbb{C}[z]^{p \times p}\) and \(\deg U(z) = n\), \(V(z) \in \mathbb{C}[z]^{p \times p}\) and \(\deg V(z) < n\), \(U(z)\) is monic.
- \(V(y_k) = U(y_k)C_k\) and \(V(z_k) = U(z_k)D_k\) for \(k = 1, \ldots, n\).

The pair \((\hat{V}(z), \hat{U}(z))\) is the only pair such that

- \(\hat{U}(z) \in \mathbb{C}[z]^{p \times p}\) and \(\deg \hat{U}(z) = n\), \(\hat{V}(z) \in \mathbb{C}[z]^{p \times p}\) and \(\deg \hat{V}(z) < n\), \(\hat{U}(z)\) is monic.
- \(\hat{V}(y_k) = C_k\hat{U}(y_k)\) and \(\hat{V}(z_k) = D_k\hat{U}(z_k)\) for \(k = 1, \ldots, n\).

Moreover, these pairs represent the same matrix rational function,

\[ [U(z)]^{-1}V(z) \equiv \hat{V}(z)[\hat{U}(z)]^{-1}. \]

Proof. See Van Barel and Vavrín [280, Theorem 3.1]. \(\square\)

As \(U(z)\) is monic, \(\det U(z) = z^n + \text{lower order terms}\) and hence \(\det U(z) \neq 0\) a.e. In other words, \(U(z)\) is invertible a.e. The same holds for \(\hat{U}(z)\).

Theorem 79. Define

\[ P(z) := \sum_{k=1}^n f_{\gamma,k}(z)P_k \]

\[ Q(z) := f_\gamma(z)I_p + \sum_{k=1}^n f_{\gamma,k}(z)P_kC_k \]

\[ \hat{P}(z) := \sum_{k=1}^n f_{\alpha,k}(z)\hat{P}_k \]

\[ \hat{Q}(z) := f_\alpha(z)I_p + \sum_{k=1}^n f_{\alpha,k}(z)D_k\hat{P}_k. \]

Then the pair \((Q(z), P(z))\) is the only pair such that

- \(Q(z) \in \mathbb{C}[z]^{p \times p}\) and \(\deg Q(z) = n\), \(P(z) \in \mathbb{C}[z]^{p \times p}\) and \(\deg P(z) < n\), \(Q(z)\) is monic.
- \(Q(y_k) = P(y_k)C_k\) and \(Q(z_k) = P(z_k)D_k\) for \(k = 1, \ldots, n\).
The pair \((\hat{Q}(z), \hat{P}(z))\) is the only pair such that

- \(\hat{Q}(z) \in \mathbb{C}[z]^{p \times p}\) and \(\deg \hat{Q}(z) = n\), \(\hat{P}(z) \in \mathbb{C}[z]^{p \times p}\) and \(\deg \hat{P}(z) < n\), \(\hat{Q}(z)\) is monic.
- \(\hat{Q}(y_k) = C_k \hat{P}(y_k)\) and \(\hat{Q}(z_k) = D_k \hat{P}(z_k)\) for \(k = 1, \ldots, n\).

Moreover, these pairs represent the same matrix rational function,

\[
|Q(z)|^{-1} P(z) \equiv \hat{P}(z)[\hat{Q}(z)]^{-1}.
\]

**Proof.** See Van Barel and Vavrín [280, Theorem 3.2].

Observe that \(Q(z)\) and \(\hat{Q}(z)\) are monic and thus invertible a.e.

Note that if \(U(y_k)\) and \(U(z_k)\), \(k = 1, \ldots, n\), are nonsingular, then the matrix rational function \([U(z)]^{-1} V(z)\) satisfies the proper rational interpolation conditions

\[
[U(y_k)]^{-1} V(y_k) = C_k \quad \text{and} \quad [U(z_k)]^{-1} V(z_k) = D_k
\]

for \(k = 1, \ldots, n\). Similarly, if \(P(y_k)\) and \(P(z_k)\), \(k = 1, \ldots, n\), are nonsingular, then the matrix rational function \([P(z)]^{-1} Q(z)\) satisfies the proper rational interpolation conditions

\[
[P(y_k)]^{-1} Q(y_k) = C_k \quad \text{and} \quad [P(z_k)]^{-1} Q(z_k) = D_k
\]

for \(k = 1, \ldots, n\). The matrix rational functions \([U(z)]^{-1} V(z)\) and \([P(z)]^{-1} Q(z)\) are different because their degree structure is different. The same conclusions hold for \(\hat{V}(z)[\hat{U}(z)]^{-1}\) and \(\hat{Q}(z)[\hat{P}(z)]^{-1}\).

An easy calculation reveals the following connections with the \(p \times p\) matrices that appear in the inversion formula (94):

\[
\begin{bmatrix}
U_1 \\
\vdots \\
U_n
\end{bmatrix} = \frac{1}{n} [D_{n,\omega} \otimes I_p] \begin{bmatrix}
U(y_1) \\
\vdots \\
U(y_n)
\end{bmatrix}, \quad \begin{bmatrix}
P_1 \\
\vdots \\
P_n
\end{bmatrix} = \frac{1}{n} [D_{n,\omega} \otimes I_p] \begin{bmatrix}
P(y_1) \\
\vdots \\
P(y_n)
\end{bmatrix}
\]

\[
\begin{bmatrix}
\hat{U}_1 \\
\vdots \\
\hat{U}_n
\end{bmatrix} = \frac{1}{n^{n-1}} [D_{n,\omega} \otimes I_p] \begin{bmatrix}
\hat{U}(z_1) \\
\vdots \\
\hat{U}(z_n)
\end{bmatrix}, \quad \begin{bmatrix}
\hat{P}_1 \\
\vdots \\
\hat{P}_n
\end{bmatrix} = \frac{1}{n^{n-1}} [D_{n,\omega} \otimes I_p] \begin{bmatrix}
\hat{P}(z_1) \\
\vdots \\
\hat{P}(z_n)
\end{bmatrix}
\]

As the points \(y\) are the \(n\)th roots of unity and \(z = \zeta y\), the right-hand sides of these formulae can be evaluated in \(O(p^2 n \log n)\) flops. The product \(x' = L^{-1} b'\) can then be calculated in \(4n p^2 + O(p n \log n)\) flops. Indeed, the matrix \(L^{-1}\) can be written as

\[
L^{-1} = \text{diag}(\hat{P}_1, \ldots, \hat{P}_n) C_P \text{diag}(U_1, \ldots, U_n) - \text{diag}(\hat{U}_1, \ldots, \hat{U}_n) C_P \text{diag}(P_1, \ldots, P_n)
\]

if \(C_P\) is given by the \(n p \times n p\) block Cauchy matrix

\[
C_P := \left[ \frac{I_p}{z_k - y_l} \right]_{k,l=1}^n
\]

Note that \(C_P = C \otimes I_p\) if \(C = C_1\) is the corresponding \(n \times n\) Cauchy matrix

\[
C = C_1 := \left[ \frac{1}{z_k - y_l} \right]_{k,l=1}^n
\]
2.3. The algorithms BlockRatInt_λ and BlockRatInt_ρ. Define \( s_k := y_k \) and \( s_{n+k} := z_k \) for \( k = 1, \ldots, n \). Define the block column vectors \( f_{\lambda,1}, \ldots, f_{\lambda,2n} \in \mathbb{C}^{2p \times p} \) as
\[
f_{\lambda,k}^T := [I_p - C_k] \quad \text{and} \quad f_{\lambda,n+k}^T := [I_p - D_k], \quad k = 1, \ldots, n.
\]
The interpolation problems that appear in the second part of Theorems 78 and 79 can be formulated as
\[
f_{\lambda,k}^TB_\lambda(s_k) = \begin{bmatrix} O_p & O_p \end{bmatrix}, \quad k = 1, \ldots, 2n,
\]
where
\[
B_\lambda(z) = \begin{bmatrix} N_\ell(z) & N_r(z) \\ D_\ell(z) & D_r(z) \end{bmatrix} \in \mathbb{C}[z]^{2p \times 2p}
\]
where \( \deg N_\ell(z) = n, \deg D_\ell(z) \leq n - 1, \deg N_r(z) \leq n - 1 \) and \( \deg D_r(z) = n \), and \( N_\ell(z) \) as well as \( D_r(z) \) monic, i.e., where \( B_\lambda(z) \) is a monic \( 2p \times 2p \) block matrix polynomial of degree \( n \). Theorems 78 and 79 assert that this problem has a unique solution given by
\[
B_\lambda^*(z) := \begin{bmatrix} \hat{Q}(z) & \hat{V}(z) \\ \hat{P}(z) & \hat{U}(z) \end{bmatrix}.
\]
Similarly, define the block column vectors \( f_{\rho,1}, \ldots, f_{\rho,2n} \in \mathbb{C}^{2p \times p} \) as
\[
f_{\rho,k} := [I_p - C_k], \quad f_{\rho,n+k} := [I_p - D_k], \quad k = 1, \ldots, n.
\]
The interpolation problems that appear in the first part of Theorems 78 and 79 can now be formulated as
\[
B_\rho(s_k)f_{\rho,k} = \begin{bmatrix} O_p \\ O_p \end{bmatrix}, \quad k = 1, \ldots, 2n,
\]
where
\[
B_\rho(z) = \begin{bmatrix} N_u(z) & D_u(z) \\ N_v(z) & D_v(z) \end{bmatrix} \in \mathbb{C}[z]^{2p \times 2p}
\]
where \( \deg N_u(z) = n, \deg D_u(z) \leq n - 1, \deg N_v(z) \leq n - 1 \) and \( \deg D_v(z) = n \), and \( N_u(z) \) as well as \( D_v(z) \) monic, i.e., where \( B_\rho(z) \) is a monic \( 2p \times 2p \) block matrix polynomial of degree \( n \). Theorems 78 and 79 assert that this problem has a unique solution given by
\[
B_\rho^*(z) := \begin{bmatrix} Q(z) & P(z) \\ V(z) & U(z) \end{bmatrix}.
\]
This corresponds to the interpolation problems that we have considered in Section 2 of Chapter 8.

The algorithms BlockRatInt_λ and BlockRatInt_ρ enable us to compute \( B_\lambda^*(z) \) and \( B_\rho^*(z) \), respectively. Note that only the second block row of \( B_\lambda^*(z) \) and the second block column of \( B_\rho^*(z) \) are needed to compute the inversion parameters.
What is the arithmetic complexity of this approach? It takes \( 2p^3n^2 + O(n) \) complex multiplications and \( 2p^3n^2 + O(n) \) complex additions to compute the second block row of \( B^*_1(z) \). The updating of the residuals costs \( 4p^3n^2 + O(n) \) complex multiplications and \( 4p^3n^2 + O(n) \) complex additions. Thus the overall cost is \( 6p^3n^2 + O(n) \) complex multiplications and \( 6p^3n^2 + O(n) \) complex additions. Similarly we need \( 6p^3n^2 + O(n) \) complex multiplications and \( 6p^3n^2 + O(n) \) complex additions to compute the second block column of \( B^*_1(z) \). Therefore, by solving the interpolation problems via BLOCKRATINT, and BLOCKRATINT, we obtain a fast block Hankel solver that has arithmetic complexity \( O(p^3n^2) \). Indeed, the transformation \( H \rightarrow L \) requires \( O(p^2n \log n) \) flops whereas the transformations \( b \rightarrow \mathbf{v} \) and \( x' \rightarrow x \) both take \( O(pn \log n) \) flops. Once the interpolation problems have been solved, the parameters that appear in the inversion formula for \( L \) can be computed in \( O(p^2n \log n) \) flops and the product \( L^{-1}b \) can be evaluated in \( 4pn^2 + O(pn \log n) \) flops. A Fortran 90 as well as a Matlab implementation are available. In the Fortran version the FFTs are calculated via FFTPACK. In the next subsection we will discuss a few numerical experiments. They show the effectiveness of our approach.

2.4. Numerical examples. A matrix-vector product involving an \( n \times n \) Hankel matrix amounts to a convolution of two vectors or, equivalently, the product of two polynomials and can thus be calculated via FFTs in \( O(n \log n) \) flops [35]. By writing an approximation \( \hat{x} \) to \( x = H^{-1}b \) as

\[
\hat{x} = \begin{bmatrix}
\hat{x}_1 \\
0 \\
\vdots \\
0 \\
\end{bmatrix}
+ \begin{bmatrix}
0 \\
\hat{x}_2 \\
\vdots \\
0 \\
\end{bmatrix}
+ \cdots 
+ \begin{bmatrix}
0 \\
\hat{x}_{(n-1)p+1} \\
\vdots \\
\hat{x}_{(n-1)p+2} \\
0 \\
\vdots \\
0 \\
\end{bmatrix}
\]

it follows that the residue \( r := b - H\hat{x} \) can be calculated in \( p^2n + O(p^2n \log n) \) flops. This implies that improving an approximation for \( x \) iteratively does not add to the \( O(p^3n^2) \) complexity of our algorithm.

In the following examples we used the Fortran 90 version of our package. The calculations were done by an IBM SP2 with machine epsilon \( \varepsilon \approx 0.12 \times 10^{-15} \) in single precision and \( \varepsilon \approx 0.2210^{-15} \) in double precision.

**Example 31.** We consider single precision \( n_p \times n_p \) real block Hankel matrices \( H_{n,p} \) having \( p \times p \) blocks whose entries are random uniformly distributed in \([-1, 1]\). The right-hand sides \( b_{n,p} \in \mathbb{R}^{n_p} \) are calculated in double precision such that

\[
x_{n,p} := H_{n,p}^{-1}b_{n,p} = \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}^T.
\]
We fix the block size $p = 3$ and let $n = 1(1)100$. Figure 5 shows the results obtained by our algorithm (before and after three steps of iterative improvement in which the residue was calculated in double precision) and the results obtained via Gaussian elimination with partial pivoting (GEPP) using the LAPACK routines CGETRF and CGETRS. We have plotted $-\log_{10}(\frac{\|\hat{x}_{n,p} - x_{n,p}\|_\infty}{\|x_{n,p}\|_\infty})$ and $\log_{10}(\kappa_\infty(H_{n,p}))$ versus $np$. Also shown is $\log_{10}(\kappa_\infty(H_{n,p}))$. The condition number was calculated via the routine CGECON.
Example 32. We consider the block Hankel matrix

$$H_{2n} := \begin{bmatrix} 1 & 2 & \cdots & n \\ 2 & 0 & & \\ \vdots & \ddots & \ddots & \vdots \\ n & 0 & \cdots & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \in \mathbb{R}^{2n \times 2n}$$

for $n = 1(1)500$. We define the right-hand side $b_{2n} = [b_1^{(2n)} \cdots b_{2n}^{(2n)}]^T$ as

$$b_{2k-1}^{(2n)} = 0, \quad b_{2k}^{(2n)} = n(n+1) - k(k-1), \quad k = 1, \ldots, n.$$ 

Then the solution to $H_{2n}x_{2n} = b_{2n}$ is given by $x_{2n} = [1 \cdots 1]^T$ as one can easily verify. Figure 6 shows the results obtained by our algorithm (before and after one step of iterative improvement). The calculations were done in double precision. Also shown is $\log_{10} \kappa_\infty(H_{2n})$. The condition number was calculated via LAPACKs routine CGECON.

3. A superfast Hankel solver

We will now return to the setting of Section 1. Let $n$ be a positive integer, let $H = H_n := [h_{k+l}]_{k,l=0}^{n-1}$ be a nonsingular $n \times n$ complex Hankel matrix and let $b \in \mathbb{C}^n$. We consider the problem of computing $x := H^{-1}b$. The fast Hankel solver that we have presented in Section 1 consists of the following components:

1. The Hankel system $Hx = b$ is transformed into a Loewner system $Lx' = b'$.

   The solution $x'$ is transformed back into $x$.

2. An explicit formula for $L^{-1}$ is available. The solution $x'$ is found as the product $L^{-1}b'$.

3. This inversion formula for $L$ contains certain parameters. These are calculated by solving two linearized rational interpolation problems.

By considering Loewner matrices that are based on roots of unity we can use FFTs to perform part 1 and 2 with arithmetic complexity $O(n \log n)$ flops. In Section 1 we have used our $O(n^2)$ algorithm RATINT to solve the interpolation problems of part 3 and hence the Hankel solver that we have presented there has an overall complexity of $O(n^2)$ flops. However, if we suppose that $n$ is a power of 2, then we can solve the interpolation problems via our divide and conquer algorithm RATINTAll. This immediately leads to a superfast Hankel solver, although only for systems whose size is a power of 2.

Superfast solvers are notoriously unstable when applied to indefinite systems. We have stabilized our algorithm in several ways. Transforming the Hankel system into an interpolation problem allows us to use pivoting at the lowest “fast” level of the interpolation algorithm. If the pivots are too small, then the corresponding interpolation conditions are handled only after the recursive “superfast” part of the algorithm. This leads to a “generically superfast” solver for indefinite Hankel systems, i.e., superfast in case the number of difficult points is small and fast otherwise. The stability can also be enhanced via iterative refinement and downdating. Iterative refinement can be applied at (several or even each) intermediate level(s), based on an inversion formula for coupled Vandermonde matrices, or at the very end, based on an inversion formula for Loewner matrices. The implementation and combination
of these tools requires the determination of several parameters that have a mutual effect on each other. How to make a good choice for these parameters is still an open problem. In our numerical experiments we tuned these parameters manually. This led to very satisfactory results.

3.1. Numerical experiments. We consider double precision $n \times n$ real Hankel matrices $H_n$ whose entries are random uniformly distributed in $[0, 1]$ with $n = 2^k$ for
$k = 1, \ldots, 17$. Note that $2^{17} = 131072$. The right-hand sides $b_n \in \mathbb{R}^n$ are calculated such that $x_n := H_n^{-1} b_n = [1 \cdots 1]^T$. The calculations were done by an IBM SP2 with machine precision $\approx 0.2210^{-15}$ in double precision.

Figure 7 shows the results obtained by our algorithm in case no iterative refinement was applied and in case up to 15 steps of iterative refinement were applied to enhance the accuracy of the computed solution to the Hankel system. As a matrix-vector product involving a Hankel matrix amounts to a convolution of two vectors or, equivalently, the product of two polynomials, the residual $r_n := b_n - H_n x_n$ can be calculated via FFT in $O(n \log n)$ flops. Therefore improving an approximation for $x_n$ iteratively does not add substantially to the $O(n \log^2 n)$ complexity of our algorithm. For each value of $k$, five Hankel matrices were considered. Let $\hat{x}_n^{(l)}$ be the approximation to the solution $x_n$ after $l$ steps of iterative improvement. We made the algorithm stop as soon as the impact of iterative improvement stagnated,

$$\frac{\|b_n - H_n \hat{x}_n^{(l)}\|_1}{\|b_n\|_1} \geq \frac{1}{2} \frac{\|b_n - H_n \hat{x}_n^{(l-1)}\|_1}{\|b_n\|_1}.$$ 

Interpolation problems of size less or equal than $2^6$ were solved by our fast-only algorithm RATINT. We experimented to find an 'optimal' subproblem size. Our algorithm performed equally well with subproblem sizes of $2^4$, $2^5$, $2^7$ or even $2^8$. Its performance seems to be rather insensitive to the exact subproblem size, as long as this is 'near' $2^6$.

Our next figures represent timings. As on computer systems measurements of very small execution times are rather inaccurate, we limit the $k$-axis to that part where the results are meaningful. This is why in the following figures $k$ does not start at one but at a larger value.
Figure 8. Execution time in seconds.

Figure 9. Magnification of the execution time.

Figure 8 shows the execution time for Gaussian elimination with partial pivoting (GEPP) (these results were calculated via the LAPACK routines ZGETRF and ZGETRS), our fast algorithm and our superfast algorithm.

Figure 9 presents the results shown in Figure 8 in a different way. It gives the magnification of the execution time. For each $k$, it tells us by which factor the execution time is to be multiplied if we go from $k - 1$ to $k$. 
Figure 10. Transformation time as percentage of the execution time in case no iterative refinement was applied.

Figure 10 shows that our algorithm spends only a small part of its execution time on the transformation from Hankel to Loewner and back. The solution of the rational interpolation problems is (obviously) the most time-consuming part of the algorithm. For each value of $k$, five Hankel matrices were considered.

The computed solution to the Hankel system can be refined iteratively. Figure 11 shows how much execution time was spent on iterative refinement as percentage of the execution time in case no iterative refinement was applied. We considered one, two, three or four steps of iterative refinement. For each value of $k$ and each number of iterative refinement steps, five Hankel matrices were considered.

4. A superfast Toeplitz solver

In the previous section we have presented a superfast solver for indefinite Hankel systems. A Hankel system is first transformed into a Loewner system and then an explicit formula for the inverse of a Loewner matrix is used. This inversion formula involves certain parameters that can be computed by solving two linearized rational interpolation problems. We will now consider Toeplitz systems instead of Hankel systems and we will use an explicit formula for the inverse of a Toeplitz matrix. In Subsection 4.1 we will write this inversion formula in such a way that it can be applied by using FFTs. This formula contains certain polynomials. In Subsection 4.2 we will show how these can be computed by solving two linearized rational interpolation problems. Numerical examples will be given in Subsection 4.3. Note that in our superfast Hankel solver the size of the Hankel matrix is limited to a power of 2 whereas the superfast Toeplitz solver that we will present in this section can handle Toeplitz systems of arbitrary size.
4.1. An inversion formula for Toeplitz matrices. Let $n$ be a positive integer, let $T = T_n := [a_{k,l}]_{k,l=0}^{n-1}$ be a nonsingular $n \times n$ complex Toeplitz matrix and let $b \in \mathbb{C}^n$. We consider the problem of computing $x := T^{-1}b$.

Let $a_n \in \mathbb{C}$ be arbitrary and let $\tilde{T} = \tilde{T}_n$ be given by the $n \times (n + 1)$ matrix

$$\tilde{T} := [a_{k,l}]_{k,l=0}^{n-1,n} = \begin{bmatrix} a_0 & a_{-1} & \cdots & a_{-n+1} & a_{-n} \\ a_1 & a_0 & & \cdots & a_{-n+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-1} & \cdots & \cdots & a_0 & a_{-1} \end{bmatrix} = \begin{bmatrix} T \\ a_n \\ \vdots \\ a_{-1} \end{bmatrix}.$$  

The polynomials $u(z)$ and $v(z)$ are called the canonical fundamental system of $T$ if

- $\tilde{T}u = e_1$ and $u_n = 0$, where $e_1 := [1 \ 0 \ \cdots \ 0]^T$,
- $\tilde{T}v = 0$ and $v_n = 1$.

In other words, $u(z)$ is a polynomial of degree $n-1$ while $v(z)$ is a monic polynomial of degree $n$ such that

$$\begin{bmatrix} a_0 & a_{-1} & \cdots & a_{-n+1} \\ a_1 & a_0 & & \cdots & a_{-n+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-1} & \cdots & \cdots & a_0 & a_{-1} \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{n-1} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$  

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure11.png}
\caption{Time spent on iterative refinement as percentage of the execution time in case no iterative refinement was applied.}
\end{figure}
and
\[
\begin{bmatrix}
  a_0 & a_{-1} & \cdots & a_{-n+1} \\
  a_1 & a_0 & \cdots & \vdots \\
  \vdots & \vdots & \ddots & a_{-1} \\
  a_{n-1} & \cdots & \cdots & a_0
\end{bmatrix}
\begin{bmatrix}
  v_0 \\
  v_1 \\
  \vdots \\
  v_{n-1}
\end{bmatrix}
= 
\begin{bmatrix}
  a_{-n} \\
  a_{-n+1} \\
  \vdots \\
  a_{-1}
\end{bmatrix}.
\]

As \( T \) is assumed to be nonsingular, these polynomials exist. Moreover, \( u(z) \) is unique whereas \( v(z) \) is uniquely determined given a particular value of \( a_{-n} \). For our purposes the specific value of \( a_{-n} \) is immaterial and thus \( v(z) \) is in fact unique up to a linear combination with \( u(z) \). Note that by cancelling the last (zero) component of \( u \), one obtains the first column of \( T^{-1} \).

**Remark.** Let \( w = [w_0, \ldots, w_{n-1}]^T \in \mathbb{C}^n \) be the last column of \( T^{-1} \). If \( T_{n-1} := [a_{-1}]_{k,l=0}^{n-2} \) is also nonsingular, then Cramer’s rule implies that \( w_{n-1} \neq 0 \) and one may choose \( v(z) = z w(z)/w_{n-1} \). This choice determines the value of \( a_{-n} \). Also, if \( T_n \) is symmetric, then \( w(z) = \tilde{u}(z) \).

The *generating function* \( M(t,s) \) of a matrix \( M = [m_{k,l}]_{k,l=0}^{p,q} \) is defined as
\[
M(t,s) := \sum_{k=0}^{p} \sum_{l=0}^{q} m_{k,l} t^k s^l.
\]

**Theorem 80.** The generating function of \( T^{-1} \) is given by
\[
T^{-1}(t,s) = \frac{u(t)\tilde{u}(s) - v(t)\tilde{v}(s)}{1 - ts}.
\]

**Proof.** See Heinig and Rost [140, p. 32].

The matrix whose generating function is given by the right-hand side of (98) is called the *Toeplitz Bezoutian* of the polynomials \( u(z) \) and \( v(z) \).

Let \( N \geq n \) be a power of 2. From the previous theorem we will now derive a formula for \( T^{-1} \) that will enable us to calculate the matrix-vector product \( T^{-1} \) in \( O(N \log N) \) floating point operations.

Define \( \omega_0, \ldots, \omega_{2N-1} \) as the \( 2N \)th roots of unity,
\[
\omega_k := \exp \left( \frac{2\pi i}{2N} k \right), \quad k = 0, 1, \ldots, 2N - 1,
\]
and let \( \omega_k^+ : = \omega_{2k} = \omega_{-k+1} \) for \( k = 0, 1, \ldots, N - 1 \). Note that \( \omega_0^+, \ldots, \omega_{N-1}^+ \) are the \( N \)th roots of unity whereas \( \omega_0^-, \ldots, \omega_{N-1}^- \) are the \( N \)th roots of \( -1 \). Let \( \eta := \exp(i/N) \). Define the matrices \( \mathcal{F}_+ \) and \( \mathcal{F}_- \) as
\[
\mathcal{F}_+ := \left[ \begin{array}{c}
\omega_k^+
\end{array} \right]_{k=0}^{N-1}, \quad \mathcal{F}_- := \left[ \begin{array}{c}
\omega_k^-
\end{array} \right]_{k=0}^{N-1}.
\]

Then \( \mathcal{F}_+ / \sqrt{N} \) is unitary and \( \mathcal{F}_- = \mathcal{F}_+ \operatorname{diag}(1, \eta, \ldots, \eta^{N-1}) \). Matrix-vector products involving \( \mathcal{F}_+ \) or \( \mathcal{F}_- \) can be evaluated with arithmetic complexity \( O(N \log N) \) via the celebrated FFT.

Let \( [T^{-1}]_N \) denote the \( N \times N \) matrix
\[
[T^{-1}]_N := \begin{bmatrix}
  T^{-1} & 0 & 0 \\
  0 & 0 & 0
\end{bmatrix} \in \mathbb{C}^{N \times N}.
\]
Define \( D := \text{diag}(\omega_0^\pm, \ldots, \omega_{N-1}^\pm) \), \( D_\pm(u) := \text{diag}(u(\omega_0^\pm), \ldots, u(\omega_{N-1}^\pm)) \) and similar for the matrices \( D_\pm(v) \).

**Theorem 81.** The matrix \( [T^{-1}]_N \) can be expressed as

\[
[T^{-1}]_N = \frac{1}{2} [D_-(u) \mathcal{F}_- \mathcal{F}_+^{-1} D_+(v) - D_-(v) \mathcal{F}_- \mathcal{F}_+^{-1} D_+(u)] \mathcal{F}_+.
\]

**Proof.** As \( \mathcal{F}_+^{-1} = \frac{1}{N} \mathcal{F}_+^H \), it follows that

\[
\mathcal{F}_- [T^{-1}]_N \mathcal{F}_+^{-1} = [c_{k,l}]_{k,l=0}^{N-1}
\]

where

\[
c_{k,l} = \frac{1}{N} T^{-1}(\omega_k^-, 1/\omega_l^+)
\]

\[
= \frac{1}{N} \left( u(\omega_k^-) \hat{u}(1/\omega_l^+) - v(\omega_k^-) \hat{v}(1/\omega_l^+) \right)
\]

\[
= \frac{1}{N} \left( \frac{u(\omega_k^-) \hat{v}(\omega_l^+) - v(\omega_k^-) \hat{u}(\omega_l^+)}{1 - \omega_k^- / \omega_l^+} \right).
\]

One can easily verify that

\[
\mathcal{F}_- \mathcal{F}_+^{-1} = \frac{2}{N} \left[ \frac{1}{1 - \omega_k^- / \omega_l^+} \right]_{k,l=0}^{N-1}.
\]

The expression for \( [T^{-1}]_N \) then follows immediately. \( \square \)

The formula in the previous theorem allows us to compute the product \( x = T^{-1} b \) in \( O(N \log N) \) flops provided that the polynomials \( u(z) \) and \( v(z) \) are known. Indeed,

\[
\begin{pmatrix} x \\ 0 \end{pmatrix} = [T^{-1}]_N \begin{pmatrix} b \\ 0 \end{pmatrix}
\]

and the multiplication by \( [T^{-1}]_N \) can be done via six \( N \)-point (inverse) FFTs and \( O(N) \) flops. For preprocessing one has to compute the values of \( u(\omega_k) \) and \( v(\omega_k) \) for \( k = 0, 1, \ldots, 2N - 1 \), which amounts to two \( 2N \)-points FFTs.

### 4.2. Interpolation Interpretation

The symbol of \( T \) is defined as the function

\[ a : \mathbb{C}_0 \to \mathbb{C} : z \mapsto a(z) := \frac{a_{-n+1}}{z^{n-1}} + \cdots + \frac{a_1}{z} + a_0 + a_1 z + \cdots + a_{n-1} z^{n-1}. \]

**Theorem 82.** Suppose \( a_{-n} = 0 \). Then the polynomials \( u(z) \) and \( v(z) \) are the canonical fundamental system of \( T \) if and only if the following linearized rational interpolation conditions are satisfied:

\[
\omega_k^u \hat{u}(\omega_k) - a(\omega_k) u(\omega_k) = 0, \quad k = 0, 1, \ldots, 2N - 1,
\]

where \( \deg \hat{u}(z) \leq 2N - n \), \( \hat{u}_{n,2N-n} = 1 \) and \( \deg u(z) \leq n \), \( u_n = 0 \) and

\[
\omega_k^v \hat{v}(\omega_k) - a(\omega_k) v(\omega_k) = 0, \quad k = 0, 1, \ldots, 2N - 1,
\]

where \( \deg \hat{v}(z) \leq 2N - n \), \( \hat{v}_{n,2N-n} = 0 \) and \( \deg v(z) \leq n \), \( v_n = 1 \).
Proof. Let $\delta \in \mathbb{C}$. Let $w(z)$ be a polynomial of degree $\leq n$ such that $\hat{T}w = \delta e_1$. The case $\delta = 1$ corresponds to $w(z) = u(z)$ whereas $\delta = 0$ corresponds to $w(z) = v(z)$. The condition $\hat{T}w = \delta e_1$ is equivalent to the existence of polynomials $r_-(z)$ and $r_+(z)$ of degree $\leq n-1$ with $r_0 = \delta$ such that

$$a(z)w(z) = r_-(1/z) + z^n r_+(z).$$

It follows that

$$a(\omega_k)w(\omega_k) = r_-(\omega_k^{-1}) + \omega_k^n r_+(\omega_k) = r_-(\omega_k^{-1}) + \omega_k^{n-2N} r_+(\omega_k)$$

for $k = 0, 1, \ldots , 2N-1$. Define

$$r(z) := r_-(z) + z^{2N-n} r_+(1/z).$$

Then $r(z)$ is a polynomial of degree $\leq 2N-n$ and $r_0 = \delta$. Thus

$$a(\omega_k)w(\omega_k) = r(\omega_k^{-1}) = \omega_k^n [\omega_k^{2N-n} r(\omega_k^{-1})] = \omega_k^n \hat{r}(\omega_k)$$

for $k = 0, 1, \ldots , 2N-1$. In other words,

$$\omega_k^n \hat{r}(\omega_k) - a(\omega_k)w(\omega_k) = 0, \quad k = 0, 1, \ldots , 2N-1,$$

where $\deg \hat{r}(z) \leq 2N-n$, $\hat{r}_{2N-n} = \delta$ and $\deg w(z) \leq n$. This proves the theorem. $\Box$

These interpolation conditions can also be written as follows:

$$\begin{bmatrix} \omega_k^n & -a(\omega_k) \end{bmatrix} B(\omega_k) = \begin{bmatrix} 0 & 0 \end{bmatrix}, \quad k = 0, 1, \ldots , 2N-1,$$

where

$$B(z) := \begin{bmatrix} \hat{r}_u(z) & \hat{r}_v(z) \\ u(z) & v(z) \end{bmatrix} \in \mathbb{C}[z]^{2 \times 2}$$

is a $2 \times 2$ matrix polynomial. The degree of the first row of $B(z)$ is equal to $2N-n$ whereas the degree of the second row of $B(z)$ is equal to $n$. Note that if $r := 2(n-N)$, then the $r$-highest degree coefficient (cf. Section 3 of Chapter 8) of $B(z)$ is equal to $I_2$. The matrix polynomial $B(z)$ can be computed via the stabilized superfast algorithm RECRATINT. The second row of $B(z)$ leads to the inversion parameters that are needed in Equation (99).

4.3. Numerical experiments. We consider double precision $n \times n$ Toeplitz matrices $T_n$ whose entries are real and random uniformly distributed in $[0,1]$ with $n = 2^k$ for $k = 1, \ldots , 18$. Note that $2^{18} = 262144$. The right-hand sides $b_n \in \mathbb{R}^n$ are calculated such that $x_n := T_n^{-1} b_n = [1 \cdots 1]^T$. The calculations were done by an IBM SP2 with machine precision $\approx 0.22 \times 10^{-15}$ in double precision.

Figures 12 and 13 show the results obtained by our algorithm in case no iterative refinement is applied (the symbols ‘+’) and in case up to 10 steps of iterative refinement are applied (the symbols ‘o’) to enhance the accuracy of the computed solution to the Toeplitz system. Interpolation problems of size less than or equal to $2^8$ are solved by our fast-only algorithm. For each value of $k$ we consider five (random) Toeplitz matrices.
Figure 12. $\frac{\|h_n - T_n x_n\|}{\|h_n\|}$ versus $k = \log_2 n$ for $k = 1, \ldots, 18$.

Figure 13. $\frac{\|h_n - T_n x_n\|}{\|h_n\|}$ versus $k = \log_2 n$ for $k = 1, \ldots, 18$.

Our next figures represent timings. As on our computer system measurements of execution times are done in units of 0.01 seconds, we limit the $k$-axis to that part where the results are meaningful. This is why in the following figures $k$ does not start at one but at a larger value.
Figure 14. Execution time in seconds.

Figure 14 shows the execution time (in seconds) for Gaussian elimination with partial pivoting (these results were calculated via the LAPACK routines ZGETRF and ZGETRS), our fast algorithm and our superfast algorithm in case no iterative refinement is applied. The results are indicated with the symbols ‘+’, ‘o’ and ‘x’, respectively.

Figure 15 presents the results shown in Figure 14 in a different way. It gives the magnification of the execution time. For each $k$, it tells us by which factor the execution time is to be multiplied if we go from $k - 1$ to $k$.

Figures 16 and 17 are related to our superfast solver. For each value of $k$ we consider five (random) Toeplitz matrices. No iterative refinement is applied. Figure 16 shows the percentage of the execution time spent to compute the input data for the interpolation problem formulated in Theorem 82, i.e., the time needed to evaluate the $a(\omega_k)$'s. Figure 17 shows the percentage of the execution time spent to apply the inversion formula given in Theorem 81 once the interpolation problem has been solved.

We also consider matrices of size $n = 10000(5000)100000$. The entries are again real and random uniformly distributed in $[0,1]$ and the right-hand sides are again calculated such that all the entries of the solution vector are equal to one. For each value of $n$ we consider five matrices. Figure 18 shows the execution time (in seconds). The results are indicated with the symbol ‘x’. The symbols ‘o’ correspond to the case where $n$ is a power of two.

One expects that for $n$ in the range $2^{k-1} < n \leq 2^k$ the execution time is more or less equal to that of the system of size $2^k$. In practice, the execution time is less. This can be explained as follows. At the lowest level some of the first interpolation problems can be solved via polynomial interpolation, i.e., by applying FFT.
The computed solution can be refined iteratively. Figure 19 shows how much execution time is spent on iterative refinement as percentage of the execution time in case no iterative refinement is applied. We consider one, two, three or four steps of iterative refinement. The results are represented by the symbols ‘x’, ‘o’, ‘+’ and
So far we have only considered iterative refinement at the Toeplitz level, i.e., we have refined the computed solution to the Toeplitz system iteratively. Iterative refinement can also be applied at the interpolation level, cf. the inversion formula.
for coupled Vandermonde matrices given in Theorem 67. In our next experiment we apply up to four steps of iterative refinement at the highest interpolation level. The timing results are shown in Figure 20. We compare the execution time spent on this kind of iterative refinement to the total execution time in case no iterative refinement whatsoever is applied. Observe that this kind of iterative refinement is much more expensive than iterative refinement applied at the Toeplitz level.

Iterative refinement at an interpolation level may be preceded by downdating. Numerical experiments indicate that the time needed to search the interpolation points that have to be downdated is approximately 45% of the time needed for one step of iterative refinement.

The following example illustrates how important it is to find the proper combination of the stabilizing techniques that we have developed. For a certain matrix of size $2^{18}$ whose entries are random uniformly distributed in the interval $[0, 1]$, we have observed the following. By applying at most 10 steps of iterative refinement on the interpolation problems of size $2^{18}$ (this is the one but highest interpolation level; remember that a matrix of size $2^{18}$ corresponds to $2^{10}$ interpolation conditions), by considering 85 difficult points and by applying iterative refinement at the Toeplitz level, we obtain an approximation for the solution whose relative residual error is $O(10^{-15})$. If we do not consider difficult points and do not use iterative refinement at the interpolation level, then the computed approximation is so bad that iterative refinement at the Toeplitz level fails. The same holds if we only apply iterative refinement on the interpolation problems of size $2^{19}$. This clearly shows the importance of combining our stabilizing tools in the correct way. One can of course apply iterative refinement on each interpolation level, but this is very costly.
Figure 20. Up to four steps of iterative refinement at the highest interpolation level. We compare the corresponding execution time to the total execution time in case no iterative refinement at all is applied.

has to find the correct balance between accuracy and cost. This will be the subject of future research.

Acknowledgements

The results in this chapter were obtained in close collaboration with Marc Van Barel. We thank Georg Heinig for providing us with references [80, 134, 135, 137, 138, 139] and for suggesting to use the inversion formula (98).
Bibliography


Bibliography

33. D. A. Bini, L. Gemignani, and B. Meini, Factorization of analytic functions by means of Koenig’s Theorem and Toeplitz computations, Unpublished manuscript. (13)


110. G. Golub and V. Olshevsky, Pivoting on structured matrices with applications, Unpublished manuscript. (154)

111. G. H. Golub, P. Milianfar, and J. Varah, A stable numerical method for inverting shape from moments, Unpublished manuscript. (17)


118. A. Gray and G. B. Mathews, A treatise on Bessel functions and their applications to physics, Dover, 1966. (98)


120. , On solving nonlinear equations with simple singularities or nearly singular solutions, SIAM Review 27 (1985), no. 4, 537–563. (124)


125. , In what sense is the rational interpolation problem well posed?, Constr. Approx. 6 (1990), no. 4, 437–450. (68, 139)


155. Y. Ikebe, Y. Kikuchi, I. Fujishiro, N. Asai, K. Takashashio, and M. Harada, The eigenvalue problem for infinite compact complex symmetric matrices with application to the numerical computation of complex zeros of $J_0(z) - iJ_1(z)$ and of Bessel functions $J_m(z)$ of any real order $m$, Linear Algebra. Appl. 194 (1993), 35-70.  
193. P. Kravanja and P. Verlinden, On the zeros of $J_n(z) + iJ_{n+1}(z)$ and $[J_{n+1}(z)]^2 - J_n(z)J_{n+2}(z)$, Submitted to IMA J. Appl. Math. \( \{xxiv, 113 \} \)


198. N. N. Lebedev, Special functions and their applications, Dover, 1972. \( \{98 \} \)

199. T.-Y. Li, On locating all zeros of an analytic function within a bounded domain by a revised Delves/Lyness method, SIAM J. Numer. Anal. 20 (1983), no. 4, 865–871. \( \{7 \} \)


202. K. Loewner, Über monotone Matrizfunktionen, Math. Z. 38 (1934), 177–216. \( \{155 \} \)


204. D. A. MacDonald, The roots of $J_0(z) - iJ_1(z) = 0$, Quart. Appl. Math. XLVII (1989), no. 2, 375–378. \( \{xxiv, 113, 119 \} \)

205. _____, The zeros of $J_1(\zeta) - J_0(\zeta)J_2(\zeta) = 0$ with an application to swirling flow in a tube, SIAM J. Appl. Math. 51 (1991), no. 1, 40–48. \( \{xxiv, 113 \} \)

206. _____, On the computation of zeros of $J_n(z) - iJ_{n+1}(z) = 0$, Quart. Appl. Math. LV (1997), no. 4, 623–638. \( \{xxiv, 113, 114, 118, 119 \} \)

207. H. Maehly and Ch. Witzgall, Tschebycheff-Approximationen in kleinen Intervallen II, Numer. Math. 2 (1960), 293–307. \( \{139 \} \)


210. Z. Mei, A special extended system and a Newton-like method for simple singular nonlinear equations, Computing 45 (1990), 157–167. \( \{124 \} \)


217. _____, A power series method for computing singular solutions to nonlinear analytic systems, Numer. Math. 63 (1992), 301–409. \( \{124 \} \)
BIBLIOGRAPHY


244. A. D. Rawlins, Note on the roots of \( f(z) = J_0(z) - iJ_1(z) \), Quart. Appl. Math. 47 (1989), no. 2, 323-324. {113}


246. , Newton's method and high order singularities, Computers Math. Appl. 5 (1979), 79-86. {124}

247. L. Reichel, Newton interpolation at Leja points, BIT 30 (1990), 332-346. {154}


249. H. E. Salzer, Note on osculatory rational interpolation, Math. Comput. 16 (1962), 486-491. {139}


253. J. Segura and A. Gil, ELF and GNOME: Two tiny codes to evaluate the real zeros of the Bessel functions of the first kind for real orders, Unpublished manuscript. {xxiii, 98}


256. V. I. Smirnov and N. A. Lebedev, Functions of a complex variable: Constructive theory, Iliffe Books, 1968. {12}


258. , Convergence of rational interpolants, Numerical Analysis: A numerical analysis conference in honour of Jean Meinguet (A. Bultheel, A. Magnus, and P. Van Dooren, eds.), Belgian Mathematical Society, December 1996, pp. 11-32. {139}


281. H. Van de Vel, A method for computing a root of a single nonlinear equation, including its multiplicity, Computing 14 (1975), 167-171. {xxv, 123, 131, 134}


284. Z. Vavrin, Inverses of Löwner matrices, Linear Algebra Appl. 63 (1984), 227-236. {xxx, 159}


299. , Algorithm 51: A reliable and numerically stable program for rational interpolation in Lagrange data, Computing 31 (1983), 209-296. {139}


301. L. Wuytack, An algorithm for rational interpolation similar to the qd-algorithm, Numer. Math. 20 (1973), 418-424. {139}


305. J.-C. Yakoubsohn, Approximating the zeros of analytic functions by the exclusion algorithm, Numerical Algorithms 6 (1994), 63-88. {13}


307. X. Ying and I. N. Katz, A reliable argument principle algorithm to find the number of zeros of an analytic function in a bounded domain, Numer. Math. 53 (1988), 143-163. {9, 11}

308. , A simple reliable solver for all the roots of a nonlinear function in a given domain, Computing 41 (1989), no. 4, 317-333. {13}

Bai/, Z/./,
Anderson/, E/.
A t anasso v a/, L/.
Bini/, D/. A/.
Belingeri/, C/.
Berntsen/, J/.
Antoulas/, A/. C/.
Ari/, H/. Lev/-/
Berr ut/, J/.-P/.
Beckermann/, B/.
Beckers/, M/.
Barnett/, A/. R/., xxiii/.
Ammar/, G/. S/.
A nr/, B/. D/. O/.
A les/, A/.
Abrahamowitz/, M/., 98, 102
Aremenberg/, I/. A/., xxv, 83, 84, 86, 88, 125
Allgower/, E/. L/., 124
Ammar/, G/. S/., 155
Anastasselou/, G/. E/., 13
Anastasselou/, E/. G/., see Ioakimidis, N/. I/, 7,
Anderson/, B/. O/., see Bitmead, R/. R/., 154
Anderson/, E/., 37, 92
Antoulas/, A/. C/., 68, 139
Ari/, H/. Lev/., see Lev-Ari, H/.
Asai/, N/., see Ikebe/, Y/., 98
Atanasova/, L/., 13
Bak/., V/. M/., 115
Boi/, Z/., see Anderson/, E/., 37, 92
Ball/, J/. A/., see Antoulas/, A/. C/., 68, 139
Barnett/, A/. R/., xxiii/, see Thompson/, I/. J/.,
102, 103
Beckermann/, B/., 18, 139
Beckers/, M/., 93
Belevitch/, V/., 139
Belingeri/, C/., 98
Bergweiler/, W/., 135
Berntsen/, J/., 93
Berr ut/, J/-P/., 68, 139
Ben/, D/. A/., xxviii/, 13, 153, 162, 173
Bois/., C/., see Anderson/, E/., 37, 92
Bitmead/, R/. R/., 154
Blanchard/, P/., 135
Bojanczyk/, A/., see Heink/, G/., 154, 189
Bojanczyk/, A/. W/., 24, 153
Boor/, C/. De/, 60
Boros/, T/., 154
Botten/, L/. C/., xiv, 6
Boutil/, T/., 11
Bouret/, E/., see Beckermann/, B/., 18
Brass/, H/., 93
Brent/, R/., 154
Brown/, J/. D/., 16
Brun/, M/. G/. de/, 139
Buckley/, A/. G/., 38
Bultheel/, A/., xvii/, 16, 20, 139
Bultheel/, A/., xxvii/, see Van Barel/, M/., 70,
72, 130, 140, 142, 145, 156, 154
Burniston/, E/. E/., 13
Cab/, S/., 24, 139, 153, 155
Cabay/, S/., see Labahn/, G/., 155
Canvy/, J/. F/., see Eibler/, L/. Z/., 85
Cardinal/, J/-P/., 13, 85
Carpen t er/, M/. P/., 6, 7
Carst ensen/, C/., 13
Carstensen/, C/., see Beckermann/, B/., see
Petkovic/, M/. S/., see Beckermann/, B/.,
12, 139
Chambers/, L/. G/., 98
Chan/, T/. F/., 153
Chen/, X/., see Nashted/, M/. Z/., 124
Cheney/, E/. W/., 68, 139
Choi/, D/., see Cab/, S/., 155
Chu/, M/. T/., see Brown/, J/. D/., 16
Chui/, C/. K/., see Cheney/, E/. W/., 68, 139
Claessens/, G/., 139
Collatz/, L/., 139
Constantinescu/, T/., see Sayed/, A/. H/., 139
Cools/, R/., xxi/, see Verschelde/, J/., see Kra vanja/,
P/., 83, 85
Córdova/, A/., 17
Craig/, M/. S/., xiv, see Botten/, L/. C/., 6
Crampin/, M/., 84
Croz/, J/. De/, see Anderson/, E/., 37, 92
Cucker/, F/., 57, 85
Cuyt/, A/., 78, 139
Davies/, B/., 7
De Boor/, C/., see Boor/, C/. De/
de Bruin/, M/. G/., see Bruin/, M/. G/. de
de Doncker-Kapenga/, E/., see Doncker-Kapenga/
E/., de
de Hoog/, F/., see Hoog/, F/. de
De Moor/, B/., see Moor/, B/. De
Decker/, D/. W/., 124
Delv/., L/. M/., xiii, 5
Dehes/, L/. M/., see Abd-Elall/, L/. F/., see Ly ness/, J/. N/., 28, 78
Demmel/, J/., see Anderson/, E/., 37, 92
VAN HENTENRYCK, P., 85
van Rossum, H., see Rossum, H. van
VANDER STRAETEN, M., xxv, 123, 131
Varah, J., see Golub, G. H., 17
VAVRIN, Z., xxx, 159
VAVRIN, Z., xxx, see Van Barel, M., 158, 159, 169-171
VERLINDEN, P., 115
Verlinden, P., xxiv, see Kravanja, P., 113
Vermout, V., see Gallivan, K. A., 154
VERSCHELDE, J., 85
VICTORY, H. D., see Neta, B., 124
VON WESTENHOLZ, C., 84
VRAHATIS, M. N., xxii, xxiii, 98, 101-103
Vrahatis, M. N., xiii, xxii, see Kavadas, D. J., see Kravanja, P., see Ragos, O., see Kravanja, P., 5, 35, 84, 97, 111
WALLIN, H., 139
Wampler, C. W., see Morgan, A. P., 124
Warner, D. D., see Gragg, W. B., 155
Wasilkowski, G. W., see Traub, J. F., 11
Watanabe, S., see Ojika, T., 124
WATSON, G. N., xxiii, 53, 98, 114, 116, 118
WEBER, H., 124
WERNER, H., 139
WERNER, H., see Collatz, L., 139
WERNER, W., see Schneider, C., see Weber, H., 124, 139
WHITLOCK, P. A., see Kalos, M. H., 93
WILKINSON, J. H., 6
Willems, J. C., see Antoulas, A. C., 68, 139
Witzgall, Ch., see Maehly, H., 139
Wong, R., 17, 91
Wöniakowski, H., see Traub, J. F., 11
WUYTACK, L., 139
WYNN, P., 139
YAKOUBSOHN, J.-CL., 13
YING, X., 9, 11, 13, 56
YPMA, T. J., 124
Yun, D., see Brent, R., 154
Yun, D. Y. Y., see Gragg, W. B., 155
Yuzhakov, A. P., xxv, see Azenberg, I. A., 83, 84, 86, 88, 125
ZAFIROPOULOS, F. A., xiii, xxii, see Vrahatis, M. N., see Kravanja, P., see Ragos, O., 5, 35, 84, 97, 98, 101, 111
ZAHAR, R. V. M., xv, 14, 16, 153
ZHA, H., see Freund, R. W., 24, 153
Curriculum Vitae

Education:

  Advisors: Prof. Dr. ir. Ann Haegemans and Prof. Dr. ir. Marc Van Barel.
  • “Burgerlijk Ingenieur” in Computer Science, K.U.Leuven, 1994 (80.0%).
    This is a 5-year programme. The degree is considered equivalent to the
degree of Master of Science in Engineering (US) or Diplom Ingenieur
(Germany).
  Thesis: “A Szegő Theory for Rational Matrix Functions” at the
Department of Computer Science, Division of Numerical Analysis and
Applied Mathematics. Advisor: Prof. Dr. Adhemar Bultheel.

Grants:

• Research assistant at the K.U.Leuven, Department of Computer Science,
  Division of Numerical Analysis and Applied Mathematics, October 1, 1998
• A PhD research grant from the Flemish Institute for the Promotion of
  Scientific and Technological Research in the Industry, October 1, 1994 –

Research Interests:

• Computational complex analysis: logarithmic residue based methods for
  computing all solutions to systems of analytic equations; computing zeros of
  analytic functions; zeros of Bessel functions.
• Structured numerical linear algebra: fast and superfast methods for solving
  systems of linear equations that have low displacement rank (for example,
  Toeplitz or Hankel systems); rational interpolation.
  • Continuous Newton methods.

Conferences that I attended:

• HERCMA 98: Hellenic European Research on Computer Mathematics and
  its Applications, September 24–26, 1998, Athens University of Economics
  and Business, Athens, Greece.
• ICCAM 98: International Congress on Computational and Applied
  Mathematics, July 27 – August 1, 1998, Katholieke Universiteit Leuven,
  Leuven, Belgium.


• **Computational Methods and Function Theory ’97**, October 13–17, 1997, Nicosia, Cyprus.


• **Constructive Complex Analysis**, November 28, 1996, Leuven, Belgium.


• **Special Functions and their Applications**, a 1-day meeting on Special Functions, Orthogonal Polynomials and Rational Approximation in honour of Prof. André Ronveaux, February 1, 1996, Leuven, Belgium.

• **Special Topics in Numerical Analysis and Applied Mathematics**, a 1-day conference in honour of Prof. Jean Meinguet, December 1, 1995, Louvain-la-Neuve, Belgium.

• **SEA-95: Résolution des Systèmes d’Équations Algébriques**, November 13–17, 1995, Luminy, Marseille, France.

• **Workshop Rational Approximation**, October 17, 1995, Leuven, Belgium.

**Conference talks:**


- P. Kravanja and M. Van Barel, *A fast Hankel solver based on an inversion formula for Loewner matrices*, DEA Course MAPA3011: Special Topics in
Numerical Linear Algebra, November 18, 1996. Institut de Mathématique Pure et Appliquée, Université Catholique de Louvain, Louvain-la-Neuve, Belgium.


The talks were given by the person whose name is underlined.

**Research stays:**

- Research stay at the Department of Mathematics of the University of Patras (Patras, Greece). March 22–April 18, 1998. Invited by Prof. M. Vrahatis.


**Research Publications:**

In international journals:


In conference proceedings:


Submitted papers:

- P. Kravanja and P. Verlinden, On the zeros of $J_n(z) \pm iJ_{n+1}(z)$ and $[J_{n+1}(z)]^2 - J_n(z)J_{n+2}(z)$. Submitted to IMA J. Appl. Math.

In preparation:

- M. Van Barel, G. Heinig and P. Kravanja, A stabilized superfast solver for indefinite Toeplitz systems.
- M. Van Barel and P. Kravanja, Generically superfast computation of Hankel determinants.

I guided the following Master's Thesis:


**Teaching Experience:** Numerical Analysis, Calculus.
Editorial Work:
- Editor of “The SCME Digest,” an electronically distributed newsletter on Scientific Computing and Mathematical Engineering.

Memberships:
- ILAS: International Linear Algebra Society

E-mail: Peter.Kravanja@na-net.ornl.gov