Chapter 1

Introduction

1.1 Finding Zeros of Polynomials

1.1.1 Iterative Methods

The identification of the zeros of a polynomial

$$p(z) := z^n + a_{n-1}z^{n-1} + \cdots + a_0, \quad a_k \in \mathbb{C}, \quad k = 0, \ldots, n-1 \tag{1.1}$$

is a fundamental task in scientific computation that arises in many problems in science and engineering. It is one of the classical mathematical problems with longest and richest history. The solution of quadratic equations was known to the Arab scholars of the early Middle Ages. The attempts to find solution formulae which would involve only arithmetic operations and radicals succeeded in the 16th century for polynomials of degrees three and four. However, N.H. Abel in the early 19th century showed that polynomials of degree five or more could not be solved by a formula involving roots of expressions in the coefficients, as those of degree up to four could be. In spite of the absence of solution formulae in radicals, the fundamental theorem of algebra states that every polynomial of degree n with complex coefficients has n roots in the complex plane. Therefore, since then researchers have concentrated on numerical methods for approximating the zeros of polynomials, and thousands of algorithms were proposed in the past four millennia (see, e.g., [33, 49]), these includes the famous Newton method of the 17th century, Bernoulli's

method of the 18th century, and Graeffe's method of the early 19th century. Since the advent of electronic computers in the 20th century, there have been a plethora of new methods proposed. These include the Jenkins–Traub method [24], Aberth's method [1], and several methods for simultaneous approximation of all the roots, starting with the Durand–Kerner method [78]. The convergence order of the Durand–Kerner method is quadratic, however, it does not work for polynomials having multiple or cluster of zeros.

1.1.2 Eigenvalue Methods

One of the most popular methods for computing the zeros of a polynomial is to determine the eigenvalues of the associated companion matrix. This approach is commonly referred to as the companion matrix method. Since a number of efficient numerical algorithms such as the QR method [7] are available for solving matrix eigenvalue problems, it is now possible to calculate the zeros both quickly and accurately. The use of a companion matrix also makes it easier to obtain error bounds for all zeros by applying Gerschgorin's theorem [71] to the companion matrix [5, 10].

For polynomial (1.1), if matrix $M \in \mathbb{C}^{n\times n}$ exists and which satisfies equation

$$\det(M - \lambda I) = (-1)^n p(\lambda), \tag{1.2}$$

where I denotes the identity matrix of $\mathbb{C}^{n\times n}$, then M is called a companion matrix of $p(z)$. Thus the problem of finding the zeros of $p(z)$ can be transformed to the eigenvalue problem for finding the eigenvalues of M.

There are many companion matrices for calculating polynomial zeros [12, 14, 15, 31, 68]. Among these, Frobenius companion matrix, which is defined directly in terms of the coefficients of the polynomial, is probably the most well known.

For polynomial (1.1), Frobenius companion matrix is defined by

$$G := \begin{pmatrix} 0 & \cdots & \cdots & 0 & -a_0 \\ 1 & \ddots & \ddots & 0 & -a_1 \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & & \ddots & 0 & -a_{n-2} \\ 0 & \cdots & & 1 & -a_{n-1} \end{pmatrix}. \tag{1.3}$$

The characteristic polynomial of G satisfies equation (1.2). The algebraic property of Frobenius companion matrix is rather moderate since it can never be Hermitain for $n > 2$, and it is normal if and only if $|a_0| = 1$ and $a_1 = a_2 = \cdots = a_{n-1} = 0$. This companion matrix is often the basis of many numerical methods for the computation of zeros of a given polynomial, and is used in the popular software package Matlab(roots function) to find the zeros of polynomials by the QR method after balancing [31]. Frobenius companion matrix uses the coefficients of the polynomial directly, this often leads to an ill-conditioning when computing the eigenvalues of the matrix by the QR method, and the zeros of the polynomial can not be calculated accurately.

1.1.3 Multiple Zeros

One of the most difficult problems in zero-finding is computing multiple zeros. It is known in classical numerical analysis theories that multiple zeros of polynomials are sensitive to perturbations and thereby ill-conditioned. A multiple zero is usually split into a cluster of zeros by perturbations in the polynomial or computational inaccuracies. As an example, consider the polynomial $p(z) = (z-1)^{10}$ which has a single 10-fold root $z = 1$. However, the Matlab function roots produces 10 scattered single roots

$$
\begin{aligned}
&1.05619886179158 + 0.01873240847737i\\
&1.05619886179158 - 0.01873240847737i\\
&1.03355773926755 + 0.04818282627243i\\
&1.03355773926755 - 0.04818282627243i\\
&0.99856054159578 + 0.05785247005951i\\
&0.99856054159578 - 0.05785247005951i\\
&0.96555483749967 + 0.04544553044827i\\
&0.96555483749967 - 0.04544553044827i\\
&0.94612801984542 + 0.01704326942574i\\
&0.94612801984542 - 0.01704326942574i
\end{aligned}
$$

If the last coefficient of polynomial $(z-1)^{100}$ is perturbed by ε. The resulting polynomial $(z-1)^{100} + \varepsilon$ has roots altered from $z = 1$ to $1 + \varepsilon^{1/100}$. With IEEE double precision

floating point arithmetic, the machine precision $\varepsilon \approx 2.2 \times 10^{-16}$, an error of $\varepsilon^{1/100} \approx 0.697$ is generated.

A number of papers were proposed for computing the multiple zeros of a polynomial [22, 34, 80]. However, there is a so-called "attainable accuracy" in calculating multiple zeros [22, 79]: if the multiplicity of a zero is m and the machine precision is k digits, then the multiple zeros can only be calculated to the accuracy of k/m correct digits. This barrier suggests the need of using multiple precision arithmetic in the computation.

Classical iterative algorithm such as Newton method and other root finders would fail to calculate multiple zeros accurately, except that a substantial extension in machine precision is made or multiple precision is used in the computation. In Chapter 3 and Chapter 4, we show that by changing the objective of computation, the multiple zeros can be calculated as simple zeros of a new polynomial and thus are not ill-conditioned and can be calculated accurately by the companion matrix method.

1.2 Finding the Zeros of Analytic Functions that Lie Inside a Circle

1.2.1 Iterative Methods and Quadrature Methods

A number of methods are available for the determination of the zeros of an analytic function $f(z)$ that lie inside a region. The best known and certainly the simplest class of methods are iterative method such as Newton method. Such methods work very well if the approximate locations of the zeros are known in advance. However, it is difficult to obtain all the zeros of $f(z)$ within a given finite region using iterative methods. Petković [53, 54, 55] presented simultaneous iterative methods for computing zeros of analytic functions. These methods can be used only if the zeros of $f(z)$ are known to be simple and if sufficiently accurate initial approximations are available. Atanassova [4] considered the case of multiple zeros but assumed that the multiplicities are known in advance.

The problem of finding the zeros of analytic functions is closely related to the computation of zeros of polynomials. One possible approach is to find a polynomial which

approximates the analytic function in the region under consideration. The polynomial may be an interpolating polynomial or simply a truncated power series. This procedure has the advantage that many methods are available to find the zeros of the polynomial. Once the zeros of the polynomial are found they may be used as starting values for an iterative process to find the corresponding zeros of $f(z)$.

Methods for the determination of zeros of analytic functions that are based on the numerical evaluations of integrals are called quadrature methods [11, 26]. Quadrature methods can be used for evaluating all the zeros of $f(z)$ in a given region. A review of such methods was given by Ioakimidis [23]. Quadrature methods calculate a polynomial $p(z)$ whose zeros coincide with the zeros of $f(z)$ in the region. In this way, the problem of finding the zeros of analytic functions is reduced to the easier problem of computing the zeros of polynomials.

1.2.2 Finding Multiple or Clusters of Zeros Using Factorization Method

If an analytic function has multiple zeros or close zeros in a given region, then the problem of evaluating all the zeros in the region by quadrature method could be very ill-conditioned. Many iterative methods for validating zeros of an analytic function require the uniqueness of a zero within a small domain. Therefore, we can not use such methods for dense clusters of zeros, because separation of simple zeros from the cluster is computationally difficult.

The multiple or close zeros of analytic function $f(z)$ can be calculated as a polynomial by factoring methods. A factoring method that finds a cluster of zeros as a polynomial factor does not need to separate simple zeros. It provides a validation method for a polynomial factor that includes all zeros in the cluster. The computation of coefficients of a polynomial of which zeros are close is more stable than the determination of locations of close zeros.

A factorization of an analytic function by reducing a problem to a solution of infinite block Toeplitz matrix is proposed in [6]. Hribernig and Stetter [21] worked on detection

and validation of clusters of zeros of polynomials. Neumaier [43] proposed a method to test the existence of root clusters and multiple roots of an analytic function, however, the derivatives of the function are needed in the computation.

For the estimation of initial approximations, global methods to find zeros or poles in a given domain [11, 29, 73] are used. When there exist dense cluster of zeros in the domain, the problem to find all the zeros in the domain is numerically extremely ill-conditioned. The location and multiplicity of a cluster of zeros in a certain domain is a stable phenomenon. Clustering methods [21, 26, 66] that find centers of clusters provide appropriate initial approximations for factoring method.

Sakurai and Sugiura [64, 65] proposed a factoring method with validation by considering a fixed point iteration for a polynomial factor $p^*(z)$ of an analytic function $f(z)$, some parameters used in the computation were assumed to be given in advance. In Chapter 5, we will present a validated method to calculate these parameters and then compute a validated polynomial factor of the analytic function $f(z)$.

1.3 Numerical Computation with Guaranteed Accuracy

1.3.1 Errors in Numerical Computations

In the course of carrying out a mathematical computation, one has to deal with the problem of errors. There are three types of errors which occur in a computation. First, there are "initial data" errors which arise when the equations of the mathematical model are formed, due to sources such as the idealistic assumptions made to simplify the model, inaccurate measurements of data, the inaccurate representation of mathematical constants. Second, there are "truncation" errors which occur when we are forced to replace an infinite process by a finite one. A typical example is the representation of a function by the first few terms of its Taylor series expansion. The third type of error is roundoff errors, which is due to the finite precision of the numbers that can be represented in a computer. Thus most numbers and the results of arithmetic operations on most numbers cannot be

represented exactly on a computer.

The unavoidability of roundoff errors in a calculation means that the exact arithmetic is impossible. That is, due to the rounding errors in finite precision arithmetic, the results obtained will not be the "true" exact values of the target quantities, but only some values that are in some sense "near" the true ones. One of the well known examples is the problem of finding the roots of the Wilkinson's polynomial

$$w(z) = (z-1)(z-2)\cdots(z-20)$$

in expanded form. The exact roots are all simple and well isolated, well the results calculated in Matlab using floating point arithmetic are

$$\begin{aligned}
&20.00033839279580, 10.00625029750179\\
&18.99702741093584, 8.99831804962360\\
&18.01178566888233, 8.00031035715436\\
&16.96952565384109, 6.99996705935380\\
&16.05089836687942, 6.00000082304303\\
&14.93191894350641, 5.00000023076008\\
&14.06837937563305, 3.99999997423112\\
&12.94716231914539, 3.00000000086040\\
&12.03450668285039, 1.99999999999934\\
&10.98361039300281, 0.99999999999983
\end{aligned}$$

The problem itself is sensitive and, no matter what method of solution is used, roundoff error is bound to introduce perturbations that will make significant changes in the solution. There is very little one can do in such cases except to identify that the situation is present and try to reformulate the overall algorithm in order to avoid the need to solve such a sensitive problem.

Since a large number of arithmetic operations is performed in the course of carrying out a computer calculation, we must be concerned with the problem of how these errors propagate and affect the final result. However, the accuracy of numeric algorithms is

notoriously hard to analyze. In practice, it is often impossible or unfeasible to predict mathematically the magnitude of the roundoff and truncation errors hidden in the output of a numeric program.

In order to be truly useful, every approximate numerical procedure must be accompanied by an accuracy specification: a statement that defines the magnitude of the errors in the output values (usually, as a function of the input values and their errors). The accuracy specification must be supported by an error analysis: a mathematical proof that the output errors obey the specifications.

Unfortunately, even for relatively simple algorithms, a rigorous error analysis is often prohibitively long, or difficult, or both [75]. Moreover, a useful accuracy specification often requires that the inputs satisfy a host of prerequisites which, in practice, are often impossible to guarantee, or even to check.

1.3.2 Error Estimation and Numerical Verification Methods

A simple and common approach for estimating the error in a floating point computation is to use more precision in the computation, and compare the results. If the results agree in many decimal places, then the computation is assumed to be correct, at least in the common part.

There have arisen several models for self validated computation [48, 38], also called automatic result verification, in which the computer itself keeps track of the accuracy of computed quantities, as part of the process of computing them. Therefore, if the magnitude of the error cannot be predicted, at least it can be known a posteriori.

The simplest and most popular of these models is R. Moore's interval arithmetic [35, 36], which we study at length in Chapter 2. The attraction of interval arithmetic is that it would not be necessary to analyse whether the conventional floating point computations are safe. As the interval results contain all possible values, a narrow interval indicates success. A wide interval does not prove that a conventionally computed result is wrong, but it does indicate a risk.

When evaluating expressions in interval arithmetic, the interval result tends to be dis-

appointingly large. This is due partly to its conservative nature: the result has to contain all possible values, including those where rounding errors combine in an unfavourable way. But the main cause of wide intervals is that interval arithmetic does not identify different occurrences of the same variable. Because it treats these as if they were unique occurrences, the result is much wider than one could reasonably expect.

Interval arithmetic has the property of correctness: result intervals are guaranteed to contain the real number that is the value of the expression. Implementations can only realize this potential by rounding floating point operations in the right direction. A basic limitation of self validated numerical models is that they can only determine the output errors a posteriori.

1.4 Organization of the Thesis

In the following we give a short description of the remaining parts of the thesis.

In Chapter 2, we give an introduction to one of the most important tools for validated computations — interval arithmetic. In this thesis, circular complex arithmetic is used in validated computations on finding zeros of analytic functions.

In Chapter 3, we consider the problem of computing the zeros of polynomials which have multiple zeros or clusters of zeros. We present an eigenvalue method by constructing a new companion matrix whose eigenvalues are simple and the zeros of the polynomial. This method can calculate the multiple zeros and their multiplicities accurately, and it is also efficient in calculating the center of the cluster of zeros and the number of zeros in the cluster.

In Chapter 4, we present a new method to compute the companion matrix presented in Chapter 3 using multiple precision arithmetic. The method uses a three-term recurrence algorithm based on Euclid's algorithm to evaluate the polynomial with arithmetic operations of $O(n^2)$, where n is the degree of the polynomial, then finds the zeros by a floating-point eigenvalue computation of the companion matrix by the QR method. Since the QR method is performed using floating point arithmetic and the other computations are performed in multiple precision arithmetic, the increase of overall computing time can

be contained. This method is efficient in computing multiple zeros or the center of the cluster of zeros of high degree polynomials.

In Chapter 5, we present a validated method to compute an upper bound of the maximum absolute value of an analytic function $f(z)$ on a circle and then perform a validated computation of the Taylor coefficients of $f(z)$. Based on these results, a factorization method is used to calculate an interval coefficient polynomial which contains a polynomial factor of $f(z)$. Circular complex arithmetic is used in the computation and the results are numerically validated.

In Chapter 6, for an analytic function $f(z)$ with a cluster of zeros around the origin, a verification method is presented to compute the number m of zeros in the cluster and compute a disk centered at the origin which contains exactly m zeros of $f(z)$. This method is based on the validated computation of the n-degree Taylor polynomial $p(z)$ of $f(z)$, and uses some well known formulae for bounding zeros of a polynomial to calculate a disk containing the cluster of zeros of $p(z)$. Rouché's theorem is used to verify that the disk contains the cluster of zeros of $f(z)$. The computations are carried out using circular complex arithmetic and the results are mathematically correct.

Finally, we conclude in Chapter 7 by summarizing the results of this thesis and giving some future research directions.