This is to certify that the

dissertation entitled

A SCALABLE ALGORITHM FOR
NON-SYMMETRIC EIGENVALUE PROBLEM

presented by

Xiaozhuo Yang

has been accepted towards fulfillment
of the requirements for

Ph.D. degree in Applied Mathematics

Tien-Yien Li
Major professor

Date May 2, 1996

A SCALABLE ALGORITHM FOR NON-SYMMETRIC EIGENVALUE PROBLEM

By

Xiaozhuo Yang

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the Degree of

DOCTOR OF PHILOSOPHY

Department of Mathematics

1996

ABSTRACT

A SCALABLE ALGORITHM FOR NON-SYMMETRIC EIGENVALUE PROBLEM

BY

Xiaozhuo Yang

This thesis is on iterative methods which find all the zeros of a monic complex polynomial simultaneously and an application to the algebraic eigenvalue problems.

In Chapter One we investigate the convergence behavior of the Durand-Kerner method on the real plane R^2 and characterize the *convergence set* completely in C^2. The Durand-Kerner method converges if and only if the initial approximations are in the convergence set.

Chapter Two presents a modified Aberth method to find the multiple zeros of a polynomial efficiently and accurately. We show that the modified Aberth method converges cubically. We also describe the property of the original Aberth method near a multiple zero and hence propose a scheme to dynamically detect the multiple zeros and their multiplicities.

In Chapter Three we apply the Aberth method, along with homotopy method, to the nonsymmetric eigenvalue problem. The algorithm has been implemented on the Intel Touchstone Delta multicomputers. The result shows this algorithm is competitive to HQR from EISPACK. A dynamical load-balancing technique is developed to balance the load across the computing nodes. The load-balanced algorithm improves the performance by up to 15%.

DEDICATION

To my wife, Ling Gao and sons, Pengling and Phillip, for their understanding, support and patience.

ACKNOWLEDGEMENTS

I am most indebted to my dissertation advisor, Professor T.Y. Li, for his encouragement, support adn advice during my graduate study at Michigan State University. His deep insight into mathematics and persistence have always influenced me.

I also would like to thank my dissertation committee members, Professor Michael Prazier, Professor Richard Hill, Professor John McCarthy, and Professor Zhengfang Zhou, for their valuable suggestions and precious time.

Contents

List of Tables

List of Figures

Chapter 1

Dynamics of the Durand-Kerner Method

1.1 Introduction

The Durand-Kerner method is an iterative method which approximates simultaneously all the n roots of a complex monic polynomial of degree n. It was first proposed by Durand [11], and later, independently, by Kerner [20]. Since then, more work on this method has appeared in the literature [17, 18, 21].

With a monic polynomial

$$p(z) = z^n + a_1 z^{n-1} + \cdots + a_{n-1} z + a_n \,, \tag{1.1}$$

the Durand-Kerner iterations are

$$z_k' = z_k - \frac{p(z_k)}{\prod_{j=1, j\neq k}^{n} (z_k - z_j)}, \quad k = 1, 2, \ldots, n \,. \tag{1.2}$$

Here, $\{z_1, z_2, \ldots, z_n\}$ is a set of approximations to the n zeros of $p(z)$. Emperatively, $\{z_1', z_2', \ldots, z_n'\}$ forms a set of better approximations.

The method was shown to have convergence order two when the polynomial has only simple roots [3, 20].

It was conjectured in [19] that this method converges for almost all initial approximations in C^n, when the initial approximations are viewed as a point $(z_1, z_2, \ldots, z_n)$ in C^n. To the best of our knowledge, this conjecture remains open, except for the case $n = 2$. It was mentioned in [19] that T. Terano gave a proof [33] for $n = 2$ in his Ph.D. thesis.

In this chapter, we investigate the convergence behavior of the Durand-Kerner method on the real plane R^2 and characterize the *convergence set* completely in C^2. The Durand-Kerner method converges if and only if the initial approximations are in the convergence set.

After the derivation of the Durand-Kerner method in Section 1.2, we discuss in Section 1.3 the convergence behavior of the Durand-Kerner method in R^2. In Section 1.4 we show that the Durand-Kerner method converges quadratically in its convergence set in C^2.

1.2 The Derivation

The Durand-Kerner method can be derived from different approaches [11, 13, 20]. Here, we give a simple derivation of the Durand-Kerner method following [2]. We also discuss some properties of the Durand-Kerner method.

Suppose that $x_1, x_2, \ldots, x_n$ are n simple roots of polynomial $p(z)$ and $z_1, z_2, \ldots, z_n$ are n initial approximations. Let $\delta_k = x_k - z_k$. Then, polynomial (1.1) can be written as

$$p(z) = \prod_{k=1}^{n} (z - (z_k + \delta_k)) .$$

Expanding the right hand side in the powers of δ_k and dropping all higher powers of δ_k yield

$$p(z) \cong \prod_{k=1}^{n}(z - z_k) - \sum_{k=1}^{n} \delta_k \prod_{j \neq k}(z - z_j)\,.$$

For a fixed index k, plugging in z_k on both sides and solving for δ_k give the first order approximation for δ_k,

$$\delta_k = -\frac{p(z_k)}{\prod_{j \neq k}(z_k - z_j)}\,. \tag{1.3}$$

This gives formula (1.2). Notice that the sum of the Durand-Kerner iterates $z_1', z_2', \ldots, z_n'$, is $-a_1$ [21].

1.3 Dynamics of the Durand-Kerner Method in R^2

In this section, we consider the Durand-Kerner method in R^2. All numbers considered are real numbers except otherwise stated. We assume that the polynomial has degree two with distinct real roots, having the following form

$$f(x) = (x - a)(x - b)\,,$$

where a and b are two distinct real numbers.

For simplicity, let $\Delta(x, y) = (x', y')$ be the Durand-Kerner iteration, where

$$\begin{cases} x' = x - \frac{f(x)}{(x-y)} \\ y' = y - \frac{f(y)}{(y-x)} \end{cases}\,. \tag{1.4}$$

Obviously, the Durand-Kerner method does not converge when the initial points are chosen from the set $D = \{(x, y) \in R^2 : x = y\}$, which makes the denominators zero. Let $J = \Delta^{-1}(D)$. This set can be described in the following theorem.

Theorem 1.3.1 *The set J has the following form*

$$J = \{(x, y) : 2xy - (a+b)(x+y) + 2ab = 0\}.$$

Moreover, the Durand-Kerner iteration maps all points in J into one single point $(\frac{a+b}{2}, \frac{a+b}{2})$.

Proof. Suppose (x, y) is in the set $\Delta^{-1}(D)$, then

$$x - \frac{(x-a)(x-b)}{x-y} = y - \frac{(y-a)(y-b)}{y-x}.$$

So,

$$\frac{x^2 - xy - x^2 + (a+b)x - ab}{x-y} = \frac{y^2 - xy - y^2 + (a+b)y - ab}{y-x}.$$

Hence,

$$2xy - (a+b)(x+y) + 2ab = 0. \tag{1.5}$$

On the other hand, let (x, y) satisfy condition (1.5). If $x \neq \frac{a+b}{2}$, then $y = \frac{(a+b)x-2ab}{2x-(a+b)}$. Bring it into the Durand-Kerner iteration (1.4), yielding

$$x' = x - \frac{(x-a)(x-b)}{x - \frac{(a+b)x-2ab}{2x-(a+b)}} = \frac{a+b}{2}.$$

Similarly,

$$y' = y - \frac{(y-a)(y-b)}{y - \frac{(a+b)y-2ab}{2y-(a+b)}} = \frac{a+b}{2}.$$

Now, we show that $x \neq \frac{a+b}{2}$ when $(x, y) \in J$.

If $x = \frac{a+b}{2}$ and $(x, y) \in J$, then

$$2\frac{a+b}{2}y - (a+b)(y + \frac{a+b}{2}) + 2ab = 0.$$

Consequently, $-\frac{(a+b)^2}{2} + 2ab = 0$. So, $(a-b)^2 = 0$. Hence $a = b$. This contradicts to our assumption that a and b are distinct.

Let $P = ((a+b)/2, (a+b)/2)$, then the Durand-Kerner iteration Δ maps J onto P. ■

Let $L = \{(x,y) : x + y = a + b\}$, a straight line on the real plane. The two fixed points $P_1 = (b,a)$ and $P_2 = (a,b)$ of the Durand-Kerner iteration function (1.4) are on L (See Figure 1.1).

Theorem 1.3.2 *After one Durand-Kerner iteration, (x', y') in (1.4) satisfies $x'+y' = a+b$.*

Proof. Choose any starting points x and y such that $x \neq y$. By the Durand-Kerner iteration (1.4),

$$\begin{aligned} x' &= x - \frac{(x-a)(x-b)}{x-y} \\ &= \frac{-xy + (a+b)x - ab}{x-y}, \end{aligned}$$

and

$$\begin{aligned} y' &= y - \frac{(y-a)(y-b)}{y-x} \\ &= \frac{-xy + (a+b)y - ab}{y-x}. \end{aligned}$$

So,

$$\begin{aligned} x' + y' - a - b &= x' - a + y' - b \\ &= \frac{-xy + (a+b)x - a(x-y) - ab}{x-y} \\ &\quad + \frac{-xy + (a+b)y - b(y-x) - ab}{y-x} \end{aligned}$$

$$= \frac{-xy + bx + ay - ab + xy - ay - bx + ab}{x - y}$$

$$= 0.$$

■

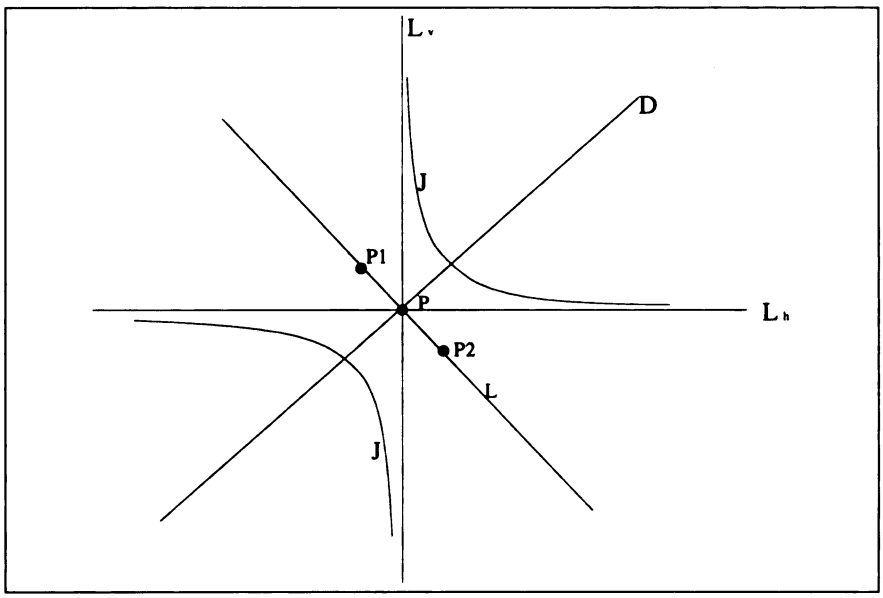


Figure 1.1: The Convergence Set

Theorem 1.3.1 and Theorem 1.3.2 show that the Durand-Kerner method maps $R^2 - D$ onto L and maps J onto point P. We will show next that $R^2 - (J \cup D)$ is the convergence set on the real plane where starting in which the Durand-Kerner method converges.

For convenience, define

$$d(x) = \frac{(a+b)x - 2ab}{2x - (a+b)}.$$

We assume $a > b$. The graph of D is a straight line passing through the point $P = ((a+b)/2, (a+b)/2)$ in R^2 while the graph of $d(x)$, the set J, has two branches (see Figure 1.1).

The function $y = d(x)$ has a horizontal asymptote and a vertical asymptote,

L_h: $y = \frac{a+b}{2}$,

L_v: $x = \frac{a+b}{2}$.

L_h and L_v divide the plane into four regions:

- $I_1 = \{(x,y) : x > \frac{a+b}{2} \text{ and } y > \frac{a+b}{2}\}$.
- $I_2 = \{(x,y) : x \leq \frac{a+b}{2} \text{ and } y \geq \frac{a+b}{2}\}$.
- $I_3 = \{(x,y) : x < \frac{a+b}{2} \text{ and } y < \frac{a+b}{2}\}$.
- $I_4 = \{(x,y) : x \geq \frac{a+b}{2} \text{ and } y \leq \frac{a+b}{2}\}$.

The graphs of J and D divide I_1 and I_3 into eight regions:

- $I_{11} = \{(x,y) \in I_1 : y < x \text{ and } y > d(x)\}$.
- $I_{12} = \{(x,y) \in I_1 : y > x \text{ and } y > d(x)\}$.
- $I_{13} = \{(x,y) \in I_1 : y > x \text{ and } y < d(x)\}$.
- $I_{14} = \{(x,y) \in I_1 : y < x \text{ and } y < d(x)\}$.
- $I_{31} = \{(x,y) \in I_3 : y > x \text{ and } y < d(x)\}$.
- $I_{32} = \{(x,y) \in I_3 : y < x \text{ and } y < d(x)\}$.
- $I_{33} = \{(x,y) \in I_3 : y < x \text{ and } y > d(x)\}$.
- $I_{34} = \{(x,y) \in I_3 : y > x \text{ and } y > d(x)\}$.

Line $y = a$ and line $x = b$ divide the regions I_2 and I_4 into eight regions:

- $I_{21} = \{(x,y) \in I_4 : b < x \leq (a+b)/2 \text{ and } a < y\}$.

- $I_{22} = \{(x,y) \in I_4 : x < b \text{ and } a < y\}$.
- $I_{23} = \{(x,y) \in I_4 : x < b \text{ and } (a+b)/2 \leq y < a\}$.
- $I_{24} = \{(x,y) \in I_4 : b < x(a+b)/2 \text{ and } (a+b)/2 < y < a\}$.
- $I_{41} = \{(x,y) \in I_4 : x > a \text{ and } b < y \leq (a+b)/2\}$.
- $I_{42} = \{(x,y) \in I_4 : (a+b)/2 < x < a \text{ and } b < y < (a+b)/2\}$.
- $I_{43} = \{(x,y) \in I_4 : (a+b)/2 \geq x < a \text{ and } y < b\}$.
- $I_{44} = \{(x,y) \in I_4 : x > a \text{ and } y < b\}$.

See Figure 1.2.

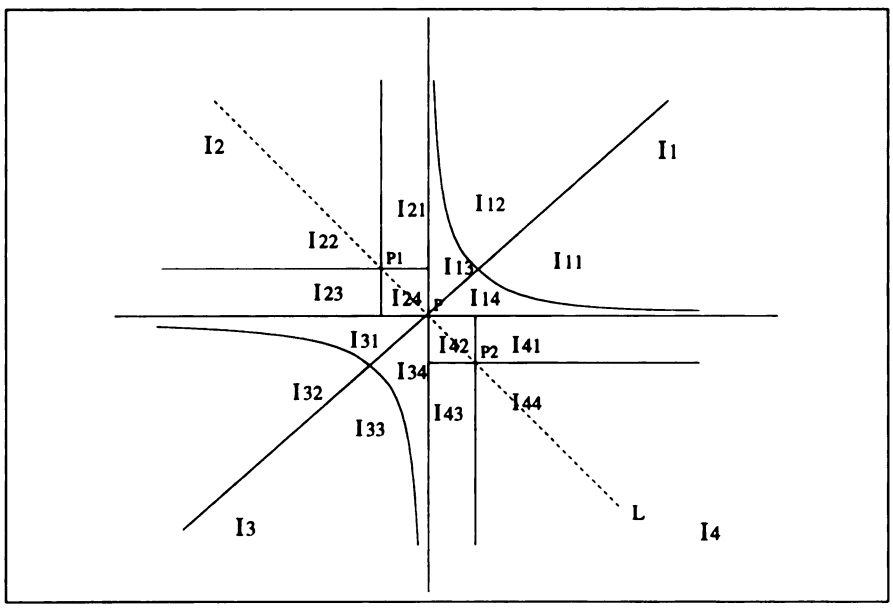


Figure 1.2: The Decomposition of R^2

The behavior of the Durand-Kerner iterations is shown in the following lemmas.

Lemma 1.3.3 $\Delta(I_{12}) \subset I_4 \cap L$.

Proof. Suppose $(x,y) \in I_{12}$, then $y > x$, $y > d(x)$ and $x > (a+b)/2$. For $(x',y') = \Delta(x,y)$, by Theorem 1.3.2, we only need to show $x' > (a+b)/2$.

$$\begin{aligned} x' - \frac{a+b}{2} &= \frac{-xy + (a+b)x - ab}{x-y} - \frac{a+b}{2} \\ &= \frac{2xy - 2(a+b)x + 2ab - (a+b)y + (a+b)x}{2(y-x)} \\ &= \frac{2xy - (a+b)x - (a+b)y + 2ab}{2(y-x)}. \end{aligned}$$

Since the denominator is positive by assumption, the numerator also needs to be positive. Now,

$$\begin{aligned} 0 &< y - d(x) \\ &= y - \frac{(a+b)x - 2ab}{2x - (a+b)} \\ &= \frac{2xy - (a+b)y - (a+b)x + 2ab}{2x - (a+b)}. \end{aligned}$$

The denominator is positive, so is the numerator. this proves $x' - \frac{(a+b)}{2} > 0$. ∎

Lemma 1.3.4 $\Delta(I_{14}) \subset I_4 \cap L$.

Proof. Suppose the initial point $(x,y) \in I_{14}$, then $y < x$, $y < d(x)$ and $x > (a+b)/2$. The assertion can be proved by applying the same arguments as in Lemma 1.3.3. ∎

Similarly, we can show that $\Delta(I_{11}) \subset I_2 \cap L$ and $\Delta(I_{13}) \subset I_2 \cap L$.

By symmetry, similar results hold when the starting points are in I_3, namely, $\Delta(I_{31}) \subset I_2 \cap L$, $\Delta(I_{33}) \subset I_2 \cap L$, $\Delta(I_{32}) \subset I_4 \cap L$ and $\Delta(I_{34}) \subset I_4 \cap L$ (see Figure 1.3).

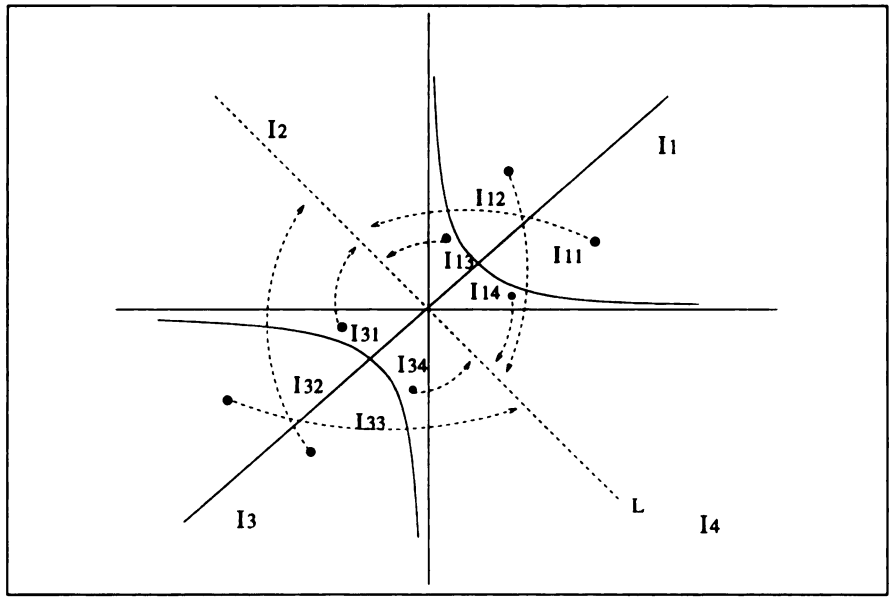


Figure 1.3: The First Iteration

Next, we show that the Durand-Kerner iterations converge to $P_2 = (a, b)$ and $P_1 = (b, a)$ when the starting points are in I_4 and I_2, respectively. By symmetry, we only consider the case of I_4 (see Figure 1.4).

Lemma 1.3.5 $\Delta(I_{41}) \subset I_{42} \cap L$.

Proof. Let $(x, y) \in I_{41}$. Notice that $y < x$, $x > a > (a+b)/2$ and $b < y < (a+b)/2$. For $(x', y') = \Delta(x, y)$,

$$\begin{aligned} x' - a &= \frac{-xy + (a+b)x - ab - a(x-y)}{x-y} \\ &= \frac{-xy + bx + ay - ab}{x-y} \\ &= \frac{(x-a)(b-u)}{x-y} \\ &< 0, \end{aligned}$$

and

$$x' - \frac{a+b}{2} = \frac{-2xy + 2(a+b)x - 2ab - (a+b)x + (a+b)y}{2(x-y)}$$

$$= \frac{-2xy + (a+b)x + (a+b)y - 2ab}{2(x-y)}.$$

Combining inequalities $\frac{(a+b)^2}{2} > 2ab$ and $x > a > \frac{(a+b)}{2}$, yields

$$\begin{aligned} -2xy + (a+b)x + (a+b)y - 2ab &> -2xy + (a+b)x + (a+b)y - \frac{(a+b)^2}{2} \\ &= 0.5((a+b) - 2y)(2x - (a+b)) \\ &> 0. \end{aligned}$$

So, $x' - \frac{a+b}{2} > 0$. This proves our assertion. ∎

Similarly, we can show that $\Delta(I_{43}) \subset I_{42} \cap L$.

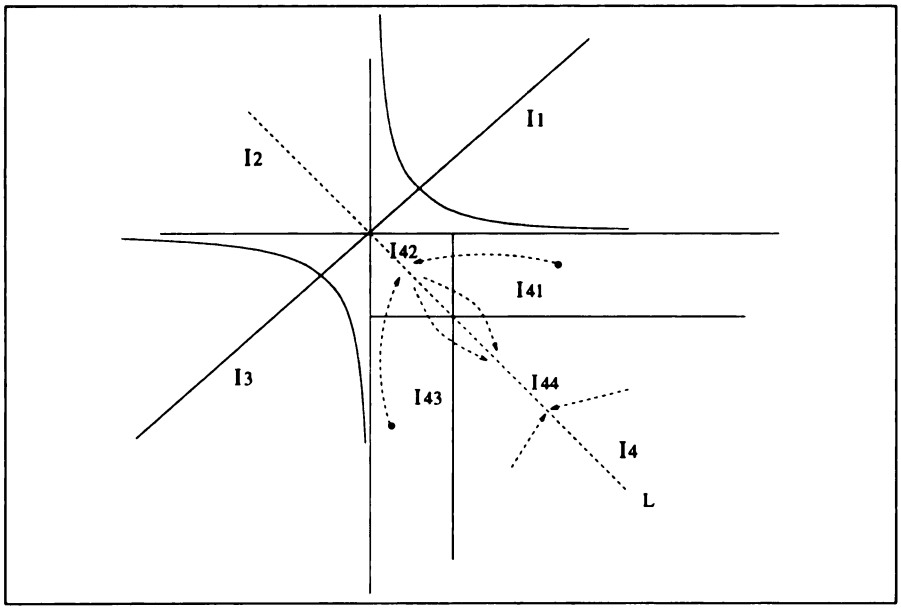


Figure 1.4: The Second Iteration

When the starting points are chosen in I_{42} we have

Lemma 1.3.6 $\Delta(I_{42}) \subset I_{44} \cap L$.

Proof. Let $(x, y) \in I_{42}$, then $y < x$, $\frac{a+b}{2} < x \leq a$ and $b \leq y < \frac{a+b}{2}$.

For $(x', y') = \Delta(x, y)$, we need to show $x' \geq a$ since (x', y') lies on L.

$$\begin{aligned} x' - a &= \frac{-xy + (a+b)x - ab - a(x-y)}{x-y} \\ &= \frac{(x-a)(b-y)}{x-y} \\ &> 0. \end{aligned}$$

■

For region I_{44}, we have

Lemma 1.3.7 $\Delta(I_{44}) \subset I_{44} \cap L$.

Proof. When $(x, y) \in I_{44}$, $x \geq a > b \geq y$. Then,

$$\begin{aligned} x' - a &= \frac{-xy + (a+b)x - ab - a(x-y)}{x-y} \\ &= \frac{(x-a)(b-y)}{x-y} \\ &\geq 0, \end{aligned}$$

and

$$\begin{aligned} y' - b &= \frac{-xy + (a+b)y - ab - b(y-x)}{y-x} \\ &= \frac{(x-a)(b-y)}{y-x} \\ &\leq 0. \end{aligned}$$

So $(x', y') \in I_{44} \cap L$. ■

We have shown that after at most two iterations the Durand-Kerner iterates lie on the line L in either I_{44} or I_{22} (see Figure 1.4).

Next, we will show that the Durand-Kerner iterations converge monotonically on the line L towards the points $P_2 = (a, b)$ and $P_1 = (b, a)$ when the initial approximations are in the region $I_{44} \cap L$ and $I_{22} \cap L$, respectively (see Figure 1.5).

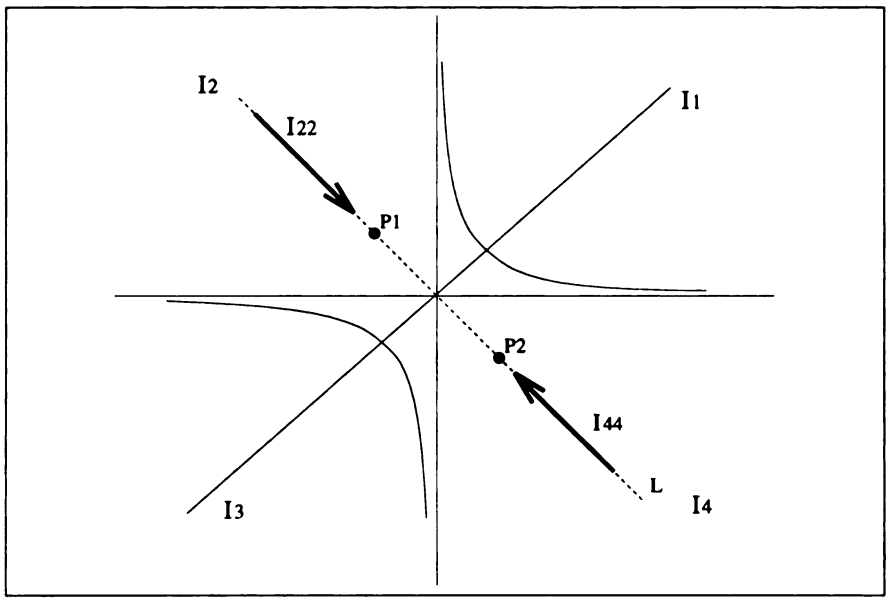


Figure 1.5: The Monotone Convergence

Theorem 1.3.8 *If the starting point* $(x, y) \in L \cap I_{44}$, *then the Durand-Kerner iterations converge to* (a, b) *monotonically. Moreover, it converges quadratically.*

Proof. Since (x, y) lies on L, we have $x + y = a + b$. For $(x', y') = \Delta(x, y)$, we also have $x' + y' = a + b$. Hence,

$$\begin{aligned} x - a &= b - y \\ y &= a + b - x\,. \end{aligned}$$

So,

$$\begin{aligned} x' - a &= \frac{(x-a)(b-y)}{x-y} \\ &= \frac{(x-a)(x-a)}{2x-(a+b)} \\ &= \frac{x-a}{2(x-\frac{a+b}{2})}(x-a) \\ &< \frac{1}{2}(x-a)\,, \end{aligned} \tag{1.6}$$

since $0 < \frac{x-a}{2(x-\frac{a+b}{2})} < 1$.

Let $(x^{(k)}, y^{(k)}) = (\Delta)^k(x,y) \equiv \Delta((\Delta)^{k-1}(x,y))$, then

$$0 < x^{(k+1)} - a < \frac{1}{2}(x^{(k)} - a) < (\frac{1}{2})^{k+1}(x-a)$$

inductively from (1.6). So, $x^{(k)}$ converges to a monotonically. At the same time, $y^{(k)}$ converges to b monotonically. Therefore, $(x^{(k)}, y^{(k)})$ converges to (a,b) monotonically.

Next, we consider the rate of convergence. Since

$$\begin{aligned}
\frac{(\|(x',y') - (a,b)\|_2)^2}{(\|(x,y) - (a,b)\|_2)^4} &= \frac{(x'-a)^2 + (y'-b)^2}{((x-a)^2 + (y-b)^2)^2} \\
&= \frac{2\frac{(x-a)^2(y-b)^2}{(x-y)^2}}{((x-a)^2 + (y-b)^2)^2} \\
&= 2\frac{(x-a)^2(x-a)^2}{(a+b-2x)^2 2(x-a)^4} \\
&= \frac{1}{(a+b-2x)^2} \\
&= \frac{1}{4}\frac{1}{(x-\frac{a+b}{2})^2} \\
&< \frac{1}{4}\frac{1}{a-\frac{a+b}{2}} \\
&= \frac{1}{(a-b)^2},
\end{aligned}$$

The Durand-Kerner method converges quadratically. ∎

Similarly, we can prove the following

Theorem 1.3.9 *If the starting point $(x,y) \in L \cap I_{22}$, then the Durand-Kerner iterations converge to (b,a). Moreover, it converges quadratically.*

In summary, after at most two iterations the Durand-Kerner method starting at

a point in $R^2 - (J \cup D)$ will converge to the solution monotonically along the line L and the rate of convergence is two.

1.4 The Complex Case

This section discusses the Durand-Kerner method in C^2. Let the two zeros a and b as well as the starting points x and y be complex numbers. Similar to the previous section, (a, b), (b, a) and (x, y) are viewed as points in two dimensional complex space C^2. Let $D = \{(x, y) \in C^2 : x = y\}$, $J = \{(x, y) \in C^2 : 2xy - (a+b)(x+y) + 2ab = 0\}$ and $L = \{(x, y) \in C^2 : x + y = a + b\}$. Notice that Theorem 1.3.1 and Theorem 1.3.2 are still true in complex case. J is the pre-image of D and after one iteration all points lie on the complex line L. We also notice that both J and D have Lebesgue measure zero in C^2.

Similar to the Durand-Kerner method on the real plane, we show that the Durand-Kerner method converges for any starting points in the set $C^2 - (J \cup D \cup S)$, where S is a set of Lebesgue measure zero in C^2.

The line L can be parameterized as follows:

$$\begin{cases} x = \frac{a+b}{2} + \frac{a-b}{2}t \\ y = \frac{a+b}{2} + \frac{b-a}{2}t \ , \end{cases} \tag{1.7}$$

where t is a complex number. When $t = 0, -1$ and 1 the corresponding points are $P = ((a + b)/2, (a + b)/2)$, $P_1 = (b, a)$ and $P_2 = (a, b)$.

Since the Durand-Kerner iterative function (1.4) maps $C^2 - (J \cup D)$ onto $L - P$, we assume, for the moment, that the starting points are chosen from $L - P$. These points can be parameterized by (1.7) with complex parameter $t \neq 0$. We will see that Durand-Kerner method does not converge when the initial approximation is chosen

by (1.7) with an imaginary number t. Let S be the set of points in C^2 corresponding to imaginary t, that is, $S = S_x \cup S_y$, where

$$S_x = \{(x,y) \in C^2 : (x-y)x - (x-a)(x-b) = (x-y)(\frac{a+b}{2} + \frac{a-b}{2}ri)\}$$

and

$$S_y = \{(x,y) \in C^2 : (y-x)y - (y-a)(y-b) = (y-x)(\frac{a+b}{2} + \frac{b-a}{2}ri)\},$$

where r is a real number and $i = \sqrt{-1}$. It is easy to see that S is a geometric surface in C^2 , hence it has Lebesgue measure zero.

Now, the Durand-Kerner iterative function (1.4) maps points in $C^2 - (J \cup D \cup S)$ to points on line L which can be parameterized by (1.7) with non-zero and non-imaginary t. Let (x,y) be on L with parametric equation (1.7). For $(x',y') = \Delta(x,y)$, we have

$$\begin{cases} x' = \frac{a+b}{2} + \frac{(a-b)}{2}\frac{(1+t^2)}{2t} \\ y' = \frac{a+b}{2} + \frac{(b-a)}{2}\frac{(1+t^2)}{2t} \end{cases} .$$

Therefore, the Durand-Kerner iterations are determined solely by the iterative behavior of the complex iterative function

$$f(z) = \frac{1+z^2}{2z}.$$

It is easy to see that $f(z)$ is just the Newton iterative function for the quadratic polynomial $q(z) = z^2 - 1$. That is, $f(z) = z - \frac{q(z)}{q'(z)}$. The Newton iterations of quadratic functions are well understood [4]. We list some results in the following

Theorem 1.4.1 *The iterations of the function $f(z)$ have the following properties:*

1. *$f^n(z)$ converges to 1 if the real part of z is positive,*

2. $f^n(z)$ converges to -1 if the real part of z is negative,

3. $f^n(z)$ does not converge if z is on the imaginary axis,

where $f^n(z) = f(f^{n-1}(z))$.

Therefore, the Durand-Kerner method converges when the starting points are chosen from $C^2 - (J \cup D \cup S)$. Since Newton's method converges quadratically, so does the Durand-Kerner method.

We conclude that the Durand-Kerner method converges if and only if the starting points are in the set $C^2 - (J \cup D \cup S)$.

Chapter 2

A Modified Aberth Method of Finding All Roots of a Polynomial with Multiple Roots

2.1 derivation

The Aberth method is a parallel iterative method for finding all roots of a monic complex polynomial with simple roots simultaneously. This method was discovered many times, see Aberth [2], Borsch-Supan [5], Docev [8], and Ehrlich [12].

For a monic complex polynomial

$$p(z) = z^n + a_1 z^{n-1} + \cdots + a_{n-1} z + a_n \ , \tag{2.1}$$

with n distinct initial approximations z_1, z_2, ..., z_n, the Aberth iterations are

$$z_k' = z_k - \frac{p(z_k)}{p'(z_k) - p(z_k) \sum_{j=1, j \neq k}^{n} \frac{1}{z_k - z_j}} \ , \quad k = 1, 2, \ldots, n. \tag{2.2}$$

It was shown [2] that the Aberth method converges cubically to the roots of $p(x)$ when it has only simple roots. It is also observed that this method converges to the n roots simultaneously for almost all initial approximations. To be more precise, considering the initial approximations $(z_1, z_2, \ldots, z_n)$ as a point in C^n, the Aberth

method converges for all initial points in $C^n - J$, where J is a subset of C^n with Lebesgue measure 0. At this stage, no proof for this observation is available.

The Aberth method is not suitable for finding the roots of a polynomial with multiple roots, since the quotient $\frac{1}{z_k - z_j}$ can be very large when two approximations z_k and z_j converge to the same multiple root. Overflow would occur. To overcome this difficulty, we propose a modified Aberth method which has the same convergence rate as the original Aberth method and can find all roots of a polynomial with multiple roots. This method reduces to the original Aberth method when the polynomial has only simple roots. The modified method, like the original method, is parallel in nature and is globally convergent in practice.

In this chapter, we first derive the modified Aberth method and describe its convergence property. In later sections we show the convergence behavior of the original Aberth method near multiple roots and discuss the issue concerning how to dynamically detect the multiple roots.

The derivation of the modified Aberth method is based on the theory of electrostatics. Let the polynomial in (2.1) have the following form

$$p(z) = (z - x_1)^{n_1}(z - x_2)^{n_2}...(z - x_m)^{n_m}\,, \tag{2.3}$$

where $x_1, x_2, \ldots, x_m$ are the m distinct roots of $p(z)$ with multiplicities $n_1, n_2, \ldots, n_m$, respectively, where $n = \sum_{k=1}^{m} n_k$.

Starting with m distinct initial approximations $z_1, z_2, \ldots, z_m$, the modified Aberth iterations are

$$z_k' = z_k - \frac{n_k p(z_k)}{p'(z_k) - p(z_k)\sum_{j=1;j\neq k}^{m} \frac{n_j}{z_k - z_j}}\,,\ k = 1, 2, \ldots, m\,. \tag{2.4}$$

We utilize the identification of complex numbers with vectors in the complex plane C. If an n_k unit plus charge is situated at the point x_k, for each k, the resulting

electromagnetic vector field at a point z is

$$\sum_{k=1}^{m} \overline{\frac{n_k}{(z - x_k)}} = \overline{\left(\frac{p'(z)}{p(z)}\right)}. \tag{2.5}$$

Given a polynomial $p(z)$, we wish to locate a root of $p(z)$ by sampling the field defined by the right hand side of (2.5) at some point z_k with multiplicity n_k, and finding the point where a single n_k unit plus charge would be located if it were causing this field. Sampling at this new point, the cycle then could be repeated. Calling the new point $z'_k = z_k + w_k$, and solving for w_k in the equation

$$\overline{\frac{n_k}{(z_k - (z_k + w_k))}} = \overline{\left(\frac{p'(z_k)}{p(z_k)}\right)},$$

leads to the modified Newton's method

$$z'_k = z_k - \frac{n_k p(z_k)}{p'(z_k)}. \tag{2.6}$$

Now, if we try to locate all roots of $p(z)$ by simultaneously applying the modified Newton's method (2.6) to m different sampling points, some of them may converge to the same (multiple) root. To avoid this, an n_k unit minus charge is assigned at the sampling point z_k. The idea is that when a sampling point z_k is near a (multiple) root, the field from the minus charge at z_k should counteract that field from the plus charge at the root, preventing a second sampling point from converging to this root. After taking conjugates, we have the iteration equation for the k-th sampling point,

$$\frac{n_k}{(z_k - (z_k + w_k))} = \frac{p'(z)}{p(z)} + \sum_{j \neq k} \frac{-n_j}{z_k - z_j}.$$

Solving for w_k yields

$$w_k = -\frac{n_k p(z_k)}{p'(z_k) - p(z_k) \sum_{j \neq k} \frac{n_j}{z_k - z_j}}. \tag{2.7}$$

This gives the modified Aberth iterative formula (2.4).

The modified Aberth iteration correction (2.7) can also be derived algebraically. Let $R_k(z) = \frac{p(z)}{\prod_{j\neq k}(z-z_j)^{n_j}}$ be a rational function, $k = 1, 2, \ldots, m$. Then R_k has the same root x_k as $p(z)$. The modified Newton's method (2.6) applied to $R_k(z)$ gives

$$\begin{aligned} w_k = & - \frac{n_k R_k(z_k)}{R_k'(z_k)} \\ = & \frac{n_k \frac{p(z_k)}{\prod_{j\neq k}(z_k-z_j)^{n_j}} (\prod_{j\neq k}(z_k - z_j)^{n_j})^2}{p'(z_k)\prod_{j\neq k}(z_k - z_j)^{n_j} - p(z_k)\frac{(\prod_{j\neq k}(z_k-z_j)^{n_j})'}{\prod_{j\neq k}(z_k-z_j)^{n_j}}}. \\ = & -\frac{n_k p(z_k)}{p'(z_k) - p(z_k)\sum_{j\neq k}\frac{n_j}{z_k-z_j}}. \end{aligned}$$

2.2 Local Convergence

In this section we discuss the local convergence property of the modified Aberth method (2.4). We will show that it converges cubically. However, when the multiplicities are estimated incorrectly in practice, only linear convergence rate can be reached.

For convenience, let

$$D_k(Z) = \sum_{j=1;j\neq k}^{m} \frac{n_j}{z_k - z_j}, \quad k = 1, 2, \ldots, m,$$

and

$$F_k(Z) = z_k - \frac{n_k p(z_k)}{p'(z_k) - p(z_k) D_k(Z)}, \quad k = 1, 2, \ldots, m\,.$$

Expanding $F_k(Z)$ in the Taylor series at the point $X = (x_1, x_2, \ldots, x_m)$ yields

$$F_k(Z) = x_k + \sum_j \frac{\partial F_k(X)}{\partial z_j}(z_j - x_j) + \frac{1}{2}\sum_{j,k=1} \frac{\partial^2 F_k(X)}{\partial z_j \partial z_k}(z_j - x_j)(z_k - x_k) + \cdots . \quad (2.8)$$

One can show that all the first and second partial derivatives are zero, so the successive iterates converge cubically. But this involves some tedious calculation of the second derivatives. Instead, we give in this section another proof of the cubical convergence. First, we calculate the first derivatives which reveal that the modified Aberth method is only linearly convergent if the multiplicities are estimated incorrectly.

Lemma 2.2.1

$$\lim_{z_k \to x_k} \frac{p(z_k)}{p'(z_k)} = 0 .$$

Proof. For simplicity, let $k = 1$.

$$\begin{aligned}
&\lim_{z_1 \to x_1} \frac{p(z_1)}{p'(z_1)} \\
= &\lim_{z_1 \to x_1} \frac{(z_1 - x_1)^{n_1}(z_1 - x_2)^{n_2} \cdots (z_1 - x_m)^{n_m}}{n_1(z_1 - x_1)^{(n_1-1)}(z_1 - x_2)^{n_2} \cdots + (z_1 - x_1)^{n_1} n_2 (z_1 - x_2)^{(n_2-1)} \cdots + \cdots} \\
= &\lim_{z_1 \to x_1} \frac{(z_1 - x_1)(z_1 - x_2)^{n_2} \cdots (z_1 - x_m)^{n_m}}{n_1(z_1 - x_2)^{n_2}(z_1 - x_3)^{n_3} \cdots + (z_1 - x_1) n_2 (z_1 - x_2)^{(n_2-1)} \cdots + \cdots} \\
= &\ 0 .
\end{aligned}$$

■

Lemma 2.2.2 *Suppose $j \neq k$, then*

$$\frac{\partial D_k(Z)}{\partial z_j} \Big|_{Z=X} = \frac{n_j}{(x_k - x_j)^2} ,$$

and

$$\frac{\partial D_k(Z)}{\partial z_k} \Big|_{Z=X} = \sum_{l \neq k} \frac{-n_k}{(x_k - x_l)^2} .$$

Proof

$$\frac{\partial D_k(Z)}{\partial z_j} \Big|_{Z=X} = \frac{n_j}{(z_k - z_j)^2} \Big|_{Z=X} = \frac{n_j}{(x_k - x_j)^2} ,$$

and

$$\frac{\partial D_k(Z)}{\partial z_k} \mid_{Z=X} = \sum_{l \neq k} \frac{-n_l}{(z_k - z_l)^2} \mid_{Z=X} = \sum_{l \neq k} \frac{-n_l}{(x_k - x_l)^2} .$$

■

Lemma 2.2.3

$$\frac{p(z)}{p'(z)} \frac{p^{(2)}(z)}{p'(z)} \mid_{z=x_k} = \frac{n_k - 1}{n_k} .$$

Proof. Write $p(z) = (z - x_k)^{n_k} g(z)$, where $g(x_k) \neq 0$. Then

$$\begin{aligned} p'(z) &= n_k(z - x_k)^{n_k - 1} g(z) + (z - x_k)^{n_k} g'(z) \\ p^{(2)}(z) &= n_k(n_k - 1)(z - x_k)^{n_k - 2} g(z) + n_k(z - x_k)^{n_k - 1} g'(z) \\ &\quad + n_k(z - x_k)^{n_k - 1} g'(z) + (z - x_k)^{n_k} g^{(2)}(z) . \end{aligned}$$

So,

$$\begin{aligned} &\frac{p(z)}{p'(z)} \frac{p^{(2)}(z)}{p'(z)} \mid_{z_k = x_k} \\ = \; &\frac{(z - x_k)^{n_k} g(z)}{n_k(z - x_k)^{(n_k - 1)} g(z) + (z - x_k)^{n_k} g'(z)} \times \\ &[\frac{n_k(n_k - 1)(z - x_k)^{n_k - 2} g(z) + n_k(z - x_k)^{n_k - 1} g'(z)}{n_k(z - x_k)^{n_k - 1} g(z) + (z - x_k)^{n_k} g'(z)} + \\ &\frac{n_k(z - x_k)^{n_k - 1} g'(z) + (z - x_k)^{n_k} g^{(2)}(z)}{n_k(z - x_k)^{n_k - 1} g(z) + (z - x_k)^{n_k} g'(z)}] \mid_{z = x_k} \\ = \; &\frac{g(z)}{n_k g(z) + (z - x_k) g'(z)} \times \\ &\frac{n_k(n_k - 1) g(z) + n_k(z - x_k) g'(z) + n_k(z - x_k) g'(z) + (z - x_k)^2 g^{(2)}(z)}{n_k g(z) + (z - x_k) g'(z)} \mid_{z = x_k} \\ = \; &\frac{g(x_k) n_k(n_k - 1) g(x_k)}{n_k g(x_k) n_k g(x_k)} \\ = \; &\frac{n_k - 1}{n_k} . \end{aligned}$$

■

Theorem 2.2.4 *All the first derivatives of $F_k(Z)$ with respect to z_l at the point X are zero, namely,*

$$\frac{\partial F_k(Z)}{\partial z_l} \mid_{Z=X} = 0\,,$$

for $k, l = 1, 2, \ldots, m$.

Proof. Notice that $p(x_k) = 0$. For $l \neq k$,

$$\begin{aligned}
\frac{\partial F_k(Z)}{\partial z_l} \mid_{Z=X} &= 0 - \frac{n_k p(z_k)\left[p(z_k)\frac{\partial D_k(Z)}{\partial z_l}\right]}{(p'(z_k) - p(z_k)D_k(Z))^2} \mid_{Z=X} \\
&= -\frac{n_k \left[\frac{p(z_k)}{p'(z_k)}\right]^2 \frac{n_k}{(z_k - z_l)^2}}{\left[1 - \frac{p(z_k)}{p'(z_k)} D_k(Z)\right]^2} \mid_{Z=X} \\
&= 0\,.
\end{aligned}$$

For $l = k$,

$$\begin{aligned}
&\frac{\partial F_k(Z)}{\partial z_k} \mid_{Z=X} \\
&= 1 - \frac{n_k p'(z_k)\left[p'(z_k) - p(z_k)D_k(Z)\right] - n_k p(z_k)\left[p^{(2)}(z_k) - p(z_k)\frac{\partial D_k(Z)}{\partial z_l} - p'(z_k)D_k(Z)\right]}{(p'(z_k) - p(z_k)D_k(Z))^2} \mid_{Z=X} \\
&= 1 - n_k \frac{\left[1 - \frac{p(z_k)}{p'(z_k)}\frac{p^{(2)}(z_k)}{p'(z_k)} + \frac{p(z_k)}{p'(z_k)}\frac{p(z_k)}{p'(z_k)}\frac{\partial D_k(Z)}{\partial z_k}\right]}{\left[1 - \frac{p(z_k)}{p'(z_k)} D_k(Z)\right]^2} \\
&= 1 - n_k \frac{\left(1 - \frac{n_k - 1}{n_k}\right)}{1} \\
&= 0\,.
\end{aligned}$$

■

Let the multiplicities $n_1, n_2, \ldots, n_m$ in the iteration formula (2.4) be replaced by estimated multiplicities $N_1, N_2, \ldots, N_m$. The modified Aberth iteration formula has

the following form:

$$z_k' = z_k - \frac{N_k p(z_k)}{p'(z_k) - p(z_k)\sum_{j=1,j\neq k}^{m} \frac{N_k}{z_k - z_j}}.$$

We then have the following convergence result.

For $l \neq k$,

$$\frac{\partial F_k(Z)}{\partial z_l} \mid_{Z=X} = -\frac{N_k \left[\frac{p(z_k)}{p'(z_k)}\right]^2 \frac{N_k}{(z_k - z_l)^2}}{\left[1 - \frac{p(z_k)}{p'(z_k)} \sum_{j=1,j\neq k}^{m} \frac{N_k}{z_k - z_j}(Z)\right]^2} \mid_{Z=X} = 0,$$

and

$$\begin{aligned}
&\frac{\partial F_k(Z)}{\partial z_k} \mid_{Z=X} \\
&= 1 - N_k \frac{\left[1 - \frac{p(z_k)}{p'(z_k)} \frac{p^{(2)}(z_k)}{p'(z_k)} + \frac{p(z_k)}{p'(z_k)} \frac{p(z_k)}{p'(z_k)} \frac{\partial(\sum_{j=1,j\neq k}^{m} \frac{N_k}{z_k - z_j})}{\partial z_k}\right]}{\left[1 - \frac{p(z_k)}{p'(z_k)} \sum_{j=1,j\neq k}^{m} \frac{N_k}{z_k - z_j}\right]^2} \mid_{Z=X} \\
&= 1 - N_k \frac{1 - \frac{n_k - 1}{n_k}}{1} \\
&= 1 - \frac{N_k}{n_k}.
\end{aligned}$$

By these results, the method converges linearly if the estimated multiplicities N_k satisfy $0 <\mid n_k - N_k \mid < n_k$, for $k = 1, 2, \ldots, m$.

When the estimated multiplicities are close (but not equal) to true multiplicities, like many other iterative methods [25], the modified Aberth method converges linearly. In this case, it does not converge cubically.

When the estimated mutiplicities are the true multiplicities, we can show the following theorem.

Theorem 2.2.5 *The modified Aberth method (2.4) converges cubically.*

Proof. Notice that $\frac{p'(z_k)}{p(z_k)} = \sum_{j\neq k} \frac{n_j}{z_k - x_j} + \frac{n_k}{z_k - x_k}$. Now,

$$\begin{aligned}
z_k' - x_k &= z_k - x_k - \frac{n_k}{\frac{p'(z_k)}{p(z_k)} - D_k(Z)} \\
&= z_k - x_k - \frac{n_k}{\sum_{j\neq k} \frac{n_j}{z_k - x_j} + \frac{n_k}{z_k - x_k} - \sum_{j\neq k} \frac{n_j}{z_k - z_j}} \\
&= z_k - x_k - \frac{n_k(z_k - x_k)}{n_k - \sum_{j\neq k} \frac{n_j(z_j - x_j)(z_k - x_k)}{(z_k - x_j)(z_k - z_j)}} \\
&= (z_k - x_k)\frac{\sum_{j\neq k} \frac{n_j(z_j - x_j)(z_k - x_k)}{(z_k - x_j)(z_k - z_j)}}{n_k - \sum_{j\neq k} \frac{n_j(z_j - x_j)(z_k - x_k)}{(z_k - x_j)(z_k - z_j)}} \\
&= \sum_{j\neq k} \frac{n_j}{(z_k - x_j)(z_k - z_j)\left[n_k - \sum_{j\neq k} \frac{n_j(z_j - x_j)(z_k - x_k)}{(z_k - x_j)(z_k - z_j)}\right]} \\
&\quad (z_k - x_k)(z_j - x_j)(z_k - x_k)\,. \qquad (2.9)
\end{aligned}$$

It is easy to see that $\frac{n_j}{(z_k - x_j)(z_k - z_j)\left[n_k - \sum_{j\neq k} \frac{n_j(z_j - x_j)(z_k - x_k)}{(z_k - x_j)(z_k - z_j)}\right]}$ in the above sum approaches $\frac{n_j}{(x_k - x_j)(x_k - x_j)n_k}$ and therefore is bounded by a positive number A as Z approaches X. Therefore, as $| z_k - x_k | < \epsilon$, $k = 1, 2, \ldots, n$, for a small number ϵ,

$$| z_k' - x_k | < A(n-1)\epsilon^3\,.$$

This shows the method converges cubically. ∎

Let $M = A(n-1)$ in the above proof. We shall estimate M in more details. Let $D(x_k, \epsilon_k) = \{z \in C : | z - x_k | < \epsilon_k\}$, $k = 1, 2, \ldots, m$, where $\epsilon_1, \epsilon_2, \ldots, \epsilon_m$ are chosen for which the disks are pairwise disjoint. The separation of the disks is measured by the minimum distance of those disks, that is,

$$\rho = min\{| z_k - z_j |, z_k \in D(x_k, \epsilon_k), z_j \in D(x_j, \epsilon_j), k \neq j\}.$$

At the r-th iteration, m disks $D^{(r)}(x_k^{(r)}, \epsilon_k^{(r)})$, $k = 1, 2, \ldots, m$, are generated by the Aberth iterations (2.4), where $\epsilon_k^{(r)} = | z_k^{(r)} - x_k |$ and $\{z_k^{(r)}\}$ are the new approximations.

Let $\rho^{(r)}$ be the separation of those disks and $\rho^{(0)} = \rho$. Intuitively, when the initial disks are chosen small enough and the initial approximations are in the initial disks, the generated disks must be nested; that is, $D^{(r+1)}(x_k^{(r+1)}, \epsilon_k^{(r+1)}) \subset D^{(r)}(x_k^{(r)}, \epsilon_k^{(r)})$.

Based on the result of Theorem 2.2.5, we have the following

Theorem 2.2.6 *Let T and t be the largest and smallest multiplicities respectively, and $\epsilon_j^{(r)}$ be the distance of the j-th root at the r-th approximation $z_j^{(r)}$ to the root x_j. Among which let $\epsilon^{(r)}$ be the largest one. Let K be an integer greater than or equal to two. If initial disks satisfy $\epsilon = \epsilon^{(0)} \leq \sqrt{\frac{t}{KT(m-1)}}\rho$, then*

$$\epsilon^{(k+1)} \leq \frac{K(m-1)T}{t\rho^2}(\epsilon^{(k)})^3 .$$

Proof. We first estimate the coefficient of $(z_k - x_k)(z_j - x_j)(z_k - x_k)$ in (2.9).

Notice that $\sum_{j\neq k} \frac{n_j(z_j-x_j)(z_k-x_k)}{(z_k-x_j)(z_k-z_j)} \geq \sum_{j\neq k} \frac{T\epsilon^2}{\rho^2} = \frac{T(m-1)\epsilon^2}{\rho^2}$. By the initial assumption, we have $\frac{T(m-1)\epsilon^2}{\rho^2} < t$. So,

$$\begin{aligned} & \left| \frac{n_j}{(z_k - x_j)(z_k - z_j)\left[n_k - \sum_{j\neq k} \frac{n_j(z_j-x_j)(z_k-x_k)}{(z_k-x_j)(z_k-z_j)}\right]} \right| \\ & \leq \frac{T}{\rho^2(t - \frac{T(m-1)}{\rho^2}\epsilon^2)} . \end{aligned}$$

Therefore,

$$\begin{aligned} \epsilon_k^{(1)} & \leq \frac{T(m-1)}{\rho^2(t - \frac{T(m-1)}{\rho^2}(\epsilon^{(0)})^2)}(\epsilon^{(0)})(\epsilon_k^{(0)})^2 \\ & \leq \frac{T(m-1)}{\rho^2(\frac{K-1}{K}t)}(\epsilon^{(0)})(\epsilon_k^{(0)})^2 \\ & = \frac{K(m-1)T}{t(K-1)\rho^2}(\epsilon^{(0)})(\epsilon_k^{(0)})^2 \\ & \leq \frac{K(m-1)T}{t\rho^2}(\epsilon^{(0)})(\epsilon_k^{(0)})^2 \end{aligned}$$

$$\leq \frac{K(m-1)T}{t\rho^2}(\epsilon^{(0)})^3.$$

Hence, $\epsilon^{(1)} \leq \frac{K(m-1)T}{t\rho^2}(\epsilon^{(0)})^3$.

Furthermore,

$$\begin{aligned}\epsilon_k^{(1)} &\leq \frac{K(m-1)T}{t\rho^2}(\epsilon^{(0)})^2(\epsilon_k^{(0)}) \\ &\leq \frac{K(m-1)T}{t\rho^2}\frac{t\rho^2}{K(m-1)T}\epsilon_k^{(0)} \\ &= \epsilon_k^{(0)}.\end{aligned}$$

Hence, each new disk is contained in the previous disk. Therefore,

$$\epsilon^{(k+1)} \leq \frac{K(m-1)T}{t\rho^2}(\epsilon^{(k)})^3$$

by induction. ∎

2.3 The Aberth Method at Multiple Zeros

Though we proved in the previous section that the modified Aberth method converges cubically, no information is available in practice whether a given polynomial has multiple roots and what the multiplicities of multiple roots are. With n initial approximations for a polynomial of degree n, one can only start with the original Aberth method. An effective program should be able to detect the multiplicities of multiple roots of a polynomial in the process of execution. We first investigate the behavior of the original Aberth method near a multiple root. In this section we show that the approximations to a multiple root become *equidistant* on the circle centered at the root. This information will be used to dynamically detect multiple roots and their multiplicities.

We first investigate the convergence behavior of the Aberth iterations applied to polynomial $p(z) = z^n$. Let $r_k = \frac{z_k}{z_1}$, $r_k' = \frac{z_k'}{z_1'}, k = 2, 3, \ldots, n$, and $r_1 = 1, r_1' = 1$.

Assume none of the approximations reaches the root of $p(z)$ in the process. Dividing both sides of (2.2) by z_1', we have

$$
\begin{aligned}
r_k' = \frac{z_k'}{z_1'} &= \frac{z_k - \frac{p(z_k)}{p'(z_k)-p(z_k)\sum_{j=1,j\neq k}^{n}\frac{1}{z_k-z_j}}}{z_1 - \frac{p(z_1)}{p'(z_1)-p(z_1)\sum_{j=2}^{n}\frac{1}{z_1-z_j}}} \\
&= \frac{\frac{z_k}{z_1} - \frac{p(z_k)}{z_1(p'(z_k)-p(z_k)\sum_{j=1,j\neq k}^{n}\frac{1}{z_k-z_j})}}{1 - \frac{p(z_1)}{z_1(p'(z_1)-p(z_1)\sum_{j=2}^{n}\frac{1}{z_1-z_j})}} \\
&= \frac{\frac{z_k}{z_1} - \frac{z_k^n}{z_1(nz_k^{n-1}-z_k^n\sum_{j=1,j\neq k}^{n}\frac{1}{z_k-z_j})}}{1 - \frac{z_1^n}{z_1(nz_1^{n-1}-z_1^n\sum_{j=2}^{n}\frac{1}{z_1-z_j})}} \\
&= \frac{r_k - \frac{r_k^n}{nr_k^{n-1}-r_k^n\sum_{j=1,j\neq k}^{n}\frac{1}{r_k-r_j}}}{1 - \frac{1}{n-\sum_{j=2}^{n}\frac{1}{1-r_j}}} \\
&= r_k\left\{\frac{1 - \frac{1}{n-r_k\sum_{j=1,j\neq k}^{n}\frac{1}{r_k-r_j}}}{1 - \frac{1}{n-\sum_{j=2}^{n}\frac{1}{1-r_j}}}\right\} \\
&\equiv Q_k(r_1, r_2, ..., r_n)\,.
\end{aligned}
$$

With $Q_1 = 1$, let $Q = (Q_1, Q_2, ..., Q_n)$. For $j = 1, 2, ..., n$, let $\omega_j = e^{\frac{2\pi(j-1)i}{n}}$ be the roots of the polynomial $P(z) = z^n - 1$, where $i = \sqrt{-1}$. It is easy to see that $(\omega_1, \omega_2, \ldots, \omega_n)$ is a fixed point of Q.

Theorem 2.3.1 *If $r = (r_1, r_2, \ldots, r_n)$ with no zero components is a fixed point of Q, then $r = \omega$.*

Proof. Under the assumption, r_k satisfies,

$$
r_k = r_k\left\{\frac{1 - \frac{r_k^{n-1}}{nr_k^{n-1}-r_k^n\sum_{j=1,j\neq k}^{n}\frac{1}{r_k-r_j}}}{1 - \frac{1}{n-\sum_{j=2}^{n}\frac{1}{1-r_j}}}\right\}, k = 1, 2, \ldots, n\,. \tag{2.10}
$$

Canceling r_k on both sides of (2.10) yields

$$1 = \frac{1 - \frac{r_k^{n-1}}{nr_k^{n-1} - r_k^n \sum_{j=1, j\neq k}^{n} \frac{1}{r_k - r_j}}}{1 - \frac{1}{n - \sum_{j=2}^{n} \frac{1}{1-r_j}}}.$$

Therefore,

$$\frac{1}{n - r_k \sum_{j=1, j\neq k}^{n} \frac{1}{r_k - r_j}} = \frac{1}{n - \sum_{j=2}^{n} \frac{1}{1-r_j}},$$

or

$$r_k \sum_{j=1, j\neq k}^{n} \frac{1}{r_k - r_j} = \sum_{j=2}^{n} \frac{1}{1 - r_j} \tag{2.11}$$

for $k = 1, 2, \ldots, n$.

We want to show that $r_1, r_2, ..., r_n$ are exactly the n roots of polynomial $z^n - 1$, which are spaced equally on the unit circle. It is easy to see that the roots of $z^n - 1$ satisfy (2.11), since $2\sum_{j\neq k} \frac{1}{r_k - r_j} = \frac{p^{(2)}(r_k)}{p'(r_k)}$ for any polynomial with all its roots being simple, where p' and $p^{(2)}$ are the first and second derivatives of p.

The right hand side of (2.11) is a constant C for any $k = 1, 2, \ldots, n$. Summing up both sides of (2.11) for k from 1 to n, the right hand side becomes nC.

Let the left hand side of (2.11) be f_n, then

$$\begin{aligned} f_n &= \sum_{k=1}^{n} r_k \sum_{j=1, j\neq k}^{n} \frac{1}{r_k - r_j} \\ &= \sum_{k=1}^{n-1} r_k \sum_{j=1, j\neq k}^{n-1} \frac{1}{r_k - r_j} + r_n \sum_{j=1}^{n-1} \frac{1}{r_n - r_j} + r_k \sum_{k=1}^{n-1} \frac{1}{r_k - r_n} \\ &= f_{n-1} + n - 1. \end{aligned} \tag{2.12}$$

Since the first term of the recursive equation is $f_2 = 1$, the solution of the recursive

relation is $f_n = \frac{n(n-1)}{2}$. Therefore, $nC = \frac{n(n-1)}{2}$. Cancelling n on both sides gives

$$r_k \sum_{j=1, j\neq k}^{n} \frac{1}{r_k - r_j} = \frac{n-1}{2}, \tag{2.13}$$

or,

$$2r_k \sum_{j=1, j\neq k}^{n} \prod_{l\neq k, j} (r_k - r_l) = (n-1) \prod_{j\neq k} (r_k - r_j). \tag{2.14}$$

Let $P(z) = \prod_{j=1}^{n}(z - r_j)$. Then (2.14) becomes

$$r_k P^{(2)}(r_k) = (n-1)P'(r_k) \tag{2.15}$$

for $k = 1, 2, ..., n$. However, the polynomials $zP^{(2)}(z)$ and $(n-1)P'(z)$ on both sides of (2.15) have the same degree $n-1$ and the same leading coefficient $(n-1)(n-2)$. They are also identical at n distinct points $r_1, r_2, \ldots, r_n$. So those two polynomials are identical. We have

$$\frac{P^{(2)}(z)}{P'(z)} = \frac{n-1}{z}.$$

Integrating both sides, we obtain $P'(z) = cz^{n-1}$ for some constant c. Hence, $P(z) = z^n - 1$ since $P(z)$ is monic and $r_1 = 1$. Therefore, $r_1, r_2, \ldots, r_n$ are the n roots of $z^n - 1 = 0$ and ω is the only fixed point of Q. ∎

Based on Theorem 2.3.1, we can show,

Theorem 2.3.2 *Applying the Aberth method (2.2) to $p(z) = z^n$, when the iterations converge to the n-ple root 0 and none of them reaches it, then they are eventually equally spaced on a circle around 0 and the modulus of each correction is $\frac{2}{n+1}$.*

Proof. The first part is a direct consequence of Theorem 2.3.1. For the second part, assume the approximations are equally spaced on a circle around 0. Without

loss of generality, let the approximations $z_1, z_2, \cdots, z_n$ be the n roots of $z^n - r = 0$ for some positive number r. Then

$$\begin{aligned} z_k' &= z_k - \frac{z_k^n}{nz_k^{n-1} - z_k^n \sum_{j \neq k} \frac{1}{z_k - z_j}} \\ &= z_k - \frac{z_k}{n - z_k \frac{n(n-1)z_k^{n-2}}{2nz_k^{n-1}}} \\ &= z_k - \frac{z_k}{n - \frac{n-1}{2}} \\ &= z_k(1 - \frac{2}{n+1}) \\ &= \frac{n-1}{n+1} z_k . \end{aligned}$$

So,

$$| z_k - z_k' | = | z_k - \frac{n-1}{n+1} z_k | = \frac{2}{n+1} | z_k | .$$

That is, the modulus of each correction is $\frac{2}{n+1}$. ∎

Similar result holds for the Aberth method applied to polynomial $p(z) = (z - x)^n$ for a complex number x.

For a general polynomial, when we analyze the behavior of the approximations converging to multiple roots, we may assume the approximations which converge to simple roots are the roots themselves, since these approximations converge to their targets very fast (we will show this later).

Theorem 2.3.3 *For a polynomial $p(z)$ of degree n which has an m-ple root x_1 $(m < n)$ and simple roots x_2, x_3, ..., x_{n-m+1}, suppose approximations z_1, z_2, ..., z_m converge to x_1 and other approximations z_{m+1}, ..., z_n take the roots of $p(z)$ other than x_1 as their initial values, then z_1, z_2, ..., z_m are eventually equally spaced on a circle around x_1 and the modulus of each correction is $\frac{2}{m+1}$.*

Proof. Since approximations $z_{m+1}, \ldots, z_n$ are roots of $p(z)$: x_2, x_3, ..., x_{n-m+1}, no iterations for them are needed. We want to show that the Aberth iterations for $z_1, z_2, \ldots, z_m$ applied to $p(z)$ are equivalent to the Aberth iterations applied to $z^m - x_1$.

Let $p(z) = (z - x_1)^m g(z)$, then the Aberth iterations applied to $p(z)$ for z_1, z_2, ..., z_m are

$$z_k' = z_k - \frac{p(z_k)}{p'(z_k) - p(z_k)\sum_{j\neq k, j=1}^{n} \frac{1}{z_k - z_j}}, \quad k = 1, 2, \ldots, m.$$

The denominators in the above iterations are

$$\begin{aligned}
& p'(z_k) - p(z_k) \sum_{j\neq k, j=1}^{n} \frac{1}{z_k - z_j} \\
= \; & m(z_k - x_1)^{m-1} g(z_k) + (z_k - x_1)^m g'(z_k) \\
& -(z_k - x_1)^m g(z_k)\Big(\sum_{j\neq k, j\leq m} \frac{1}{z_k - z_j} + \sum_{j=2}^{n-m+1} \frac{1}{z_k - x_j} \Big) \\
= \; & m(z_k - x_1)^{m-1} g(z_k) - (z_k - x_1)^m g(z_k) \sum_{j\neq k, j\leq m} \frac{1}{z_k - z_j} \\
& +(z_k - x_1)^m \left[g'(z_k) - g(z_k) \sum_{j=2}^{n-m+1} \frac{1}{z_k - x_j} \right] \\
= \; & g(z_k) \left[m(z_k - x_1)^{m-1} - (z_k - x_1)^m \sum_{j\neq k, j\leq m} \frac{1}{z_k - z_j} \right] + (z_k - x_1)^m \left[g'(z_k) - g'(z_k) \right] \\
= \; & g(z_k) \left[m(z_k - x_1)^{m-1} - (z_k - x_1)^m \sum_{j\neq k, j\leq m} \frac{1}{z_k - z_j} \right].
\end{aligned}$$

Hence,

$$\begin{aligned}
z_k' &= z_k - \frac{(z_k - x_1)^m g(z_k)}{g(z_k)\left[m(z_k - x_1)^{m-1} - (z_k - x_1)^m \sum_{j\neq k, j\leq m} \frac{1}{z_k - z_j} \right]} \\
&= z_k - \frac{(z_k - x_1)^m}{m(z_k - x_1)^{m-1} - (z_k - x_1)^m \sum_{j\neq k, j\leq m} \frac{1}{z_k - z_j}}.
\end{aligned}$$

This is exactly the Aberth method applied to $z^m - x_1$. ∎

Corollary 2.3.4 *Under the same condition as in the above theorem, let* $w_k^{(r)}$ *be the correction at the r-th iteration for the k-th approximation* $z_k^{(r)}$ *which converges to an m-ple root . Then,*

$$\lim_{r\to\infty} \frac{\mid w_k^{(r+1)} \mid}{\mid w_k^{(r)} \mid} = \frac{m-1}{m+1}.$$

Proof. By Theorem 2.3.3, we have $w_k^{(r)} = \frac{2}{m+1} z_k^{(r-1)}$ and $z_{k+1}^{(r)} = \frac{m-1}{m+1} z_k^{(r-1)}$. So $w_k^{(r+1)} = \frac{2}{m+1} z_k^{(r)} = \frac{2}{m+1}\frac{m-1}{m+1} z_k^{(r-1)}$. ∎

The above results also show that the original Aberth method converges linearly if the polynomial has multiple roots. That is, those approximations which converge to multiple roots converge to their targets linearly. Next, we show that the approximations which converge to simple roots converge much faster.

Lemma 2.3.5 *Let* $z_k = \alpha + r_k e^{\theta_k i}$, $k = 0, 1, 2, ..., m-1$, *where* $i = \sqrt{-1}$, α *is a complex number and* $r_k > 0$. *Let* $r \geq r_k$ *and* $z \in C$ *with* $\mid z - \alpha \mid > r$. *Then*

$$\sum_{k=0}^{m-1} \frac{1}{z - z_k} - \frac{m}{z-\alpha} = r\beta$$

with some β *satisfying* $\mid \beta \mid \leq \frac{m}{(\mid z-\alpha \mid)(\mid z-\alpha \mid - r)}$.

Proof.

$$\begin{aligned} \frac{1}{z - z_k} - \frac{1}{z-\alpha} &= \frac{1}{z - \alpha - r_k e^{\theta_k i}} - \frac{1}{z-\alpha} \\ &= r\frac{r_k e^{\theta_k i}}{r(z - \alpha - r_k e^{\theta_k i})(z-\alpha)}, \end{aligned}$$

and

$$\mid \frac{r_k e^{\theta_k i}}{r(z - \alpha - r_k e^{\theta_k i})(z-\alpha)} \mid \leq \frac{1}{(\mid z-\alpha \mid)(\mid z-\alpha \mid - r)}.$$

Let

$$\beta = \sum_{k=0}^{m-1} \frac{r_k e^{\theta_k i}}{r(z - \alpha - r_k e^{\theta_k i})(z - \alpha)},$$

then, $\sum_{k=0}^{m-1} \frac{1}{z-z_k} - \frac{m}{z-\alpha} = r\beta$ and $| \beta | \leq \frac{m}{(|z-\alpha|)(|z-\alpha|-r)}$. ■

Theorem 2.3.6 *Suppose the Aberth approximations applied to $p(z)$ converges and the approximation starting at z_i converges to a simple root x_i, then, eventually,*

$$| z_i^{(r+1)} - x_i | \leq C\epsilon^3,$$

where $z_i^{(r+1)}$ is the approximation at step $(r+1)$, C is a constant and

$$\epsilon = max\{| z_j^{(r)} - x_j |, j = 1, 2, \ldots, n\}.$$

Proof. Suppose $p(z) = \prod_{j=1}^{m}(z - x_j)^{n_j}$, where $x_1, x_2, ..., x_m$ are m distinct roots with multiplicities $n_1, n_2, ..., n_m$. Then

$$\frac{p'(z)}{p(z)} = \sum_{j=1}^{m} \frac{n_j}{z - x_j}.$$

Let ϵ be a small number such that the distance of every Aberth approximation to its target is decreasing as soon as all the approximations are within ϵ distance from the roots. Let ρ be the minimum distance of any two distinct roots x_k and x_j. Let ϵ be far less than ρ, say $4\epsilon < \rho$. For simplicity, let $n_1 = 1$, z_1 approximates x_1, $z_2, \ldots, z_{n_1+n_2}$ approximate $x_2, \ldots, z_{n_1+n_2+\ldots+n_{m-1}}, \ldots, z_n$ approximate x_m. Suppose all the approximations are within ϵ distance from their targets. From Lemma 2.3.5, we have,

$$\sum_{k=1}^{n_j} \frac{1}{z_1 - z_{n_1+\ldots+n_{j-1}+k}} = \frac{n_j}{z_1 - x_j} + r_j\beta_j$$

with $r_j < \epsilon$ and $| \beta_j | < \frac{n_j}{\frac{3}{4}\rho\frac{2}{4}\rho} = \frac{8n_j}{3\rho}$ since $z_{n_1+\ldots+n_{j-1}+1}, \ldots, z_{n_1+\ldots+n_j}$ converge to x_j.

So,

$$
\begin{aligned}
z_1' - x_1 &= z_1 - x_1 - \frac{1}{\frac{p'(z_1)}{p(z_1)} - \sum_{j=2}^{n} \frac{1}{z_1 - z_j}} \\
&= z_1 - x_1 - \frac{1}{\sum_{j=2}^{m} \frac{n_j}{z_1 - x_j} + \frac{1}{z_1 - x_1} - \sum_{j=2}^{m} \left(\frac{n_j}{z_1 - x_j} + r_j \beta_j \right)} \\
&= z_1 - x_1 - \frac{1}{\frac{1}{z_1 - x_1} - \sum_{j=2}^{m} r_j \beta_j} \\
&= z_1 - x_1 - \frac{z_1 - x_1}{1 - \sum_{j=2}^{m} (z_1 - x_j) r_j \beta_j} \\
&= (z_1 - x_1) \left[1 - \frac{1}{1 - \sum_{j=2}^{m} (z_1 - x_1) r_j \beta_j} \right] \\
&= (z_1 - x_1) \frac{\sum_{j=2}^{m} (z_1 - x_1) r_j \beta_j}{1 - \sum_{j=2}^{m} (z_1 - x_1) r_j \beta_j} .
\end{aligned}
$$

Hence, when $r_j < \epsilon$ we have

$$
| z_1 - x_1 | \leq \frac{\frac{8(n-1)}{3\rho}}{1 - \epsilon^2 \frac{8(n-1)}{3\rho}} \epsilon^3 .
$$

Therefore, $| z_1 - x_1 | < \frac{16(n-1)}{3\rho} \epsilon^3$ when ϵ satisfies the extra condition $\epsilon^2 < \frac{3\rho}{16(n-1)}$. ■

This theorem does not imply that the rate of convergence for the approximations which converge to simple roots is cubic. When ϵ is the largest error of all the errors $| z_k^{(r)} - x_k |$, $k = 1, 2, \ldots, m$, at step r, the conclusion of the theorem indicates the error at step $r + 1$ for approximations converging to simple roots is less than $C\epsilon^3$, for some fixed constant C. Apparently, it is much faster than the convergence rate of the approximations which converge to multiple roots where the error at step $r + 1$ is less than $C_1 \epsilon$ with $C_1 < 1$. In practice, the approximations converging to simple roots converge to the roots very fast while the approximations converging to multiple roots display a very slow convergence rate.

In [29] and [17], finding multiple roots of polynomials by Durand-Kerner method is discussed. Like the Aberth method, the Durand-Kerner method can find all the roots of a polynomial with simple roots very efficiently. It converges quadratically in this case while it converges only linearly when multiple roots exist.

In [29] a result similar to Theorem 2.3.2 for the Durand-Kerner method is proved without using the modified Durand-Kerner method to refine the approximations. In [17] a different technique is used to handle the multiple roots and obtains superlinear convergence rate. Their numerical results show that both approaches can not find the multiple roots with high accuracy.

2.4 Dynamic Estimation of Multiplicities

For a given polynomial $p(z)$ with no information in advance, one can only use the original Aberth method (2.2) in the beginning. It is desirable that the multiplicities of multiple zeros of $p(z)$ can be dynamically detected in the process of execution. The approximations can then be grouped into different groups according to the data available concerning whether they converge to the same root. Then, the modified Aberth method is applied. For a group of approximations converging to the same root, the proposed algorithm takes the average value of all the members in the group. This average value should be closer to the multiple root than any approximation in this group as the approximations are spaced equally on a circle around the root by Theorem 2.3.2 and Theorem 2.3.3. This average is used in the modified Aberth iterations. The same approach can be found in [17] which uses these properties without proof. Each z_k takes a small random perturbation of the averaged value for the next iteration since the same Durand-Kerner iteration functions are used and the approximations must be distinct. While this approach provides a supper linear

convergence rate, it does not yield a high accuracy.

In [29], the correction

$$-w_k = -\frac{f(z_k)}{\prod_{j\neq k}(z_k - z_j)}$$

is multiplied by n_k, the estimated multiplicity, to be added to z_k, since the modulus of the correction is $\frac{1}{n_k}$ for the Durand-Kerner method. In this way, quadratic convergence is achieved theoretically. Again, this approach can not yield results with high accuracy since no new formulae are used in the process of iterations. None of the remedies can avoid the overflow problem because the reciprocals of small differences of close approximations are too large. The modified Aberth method introduced in this chapter can avoid the overflow problem because the approximations in the same group are treated as a single value in the new formula. Consequently, the roots of the polynomial can be obtained with high accuracy.

Our technique of dividing approximations into groups is based on the following observations:

1. If the approximations to an m-ple root are sufficiently close, m approximations are situated on a small circle centering around the root by an angle of $\frac{2\pi}{m}$.

2. Every correction directs toward the center of the circle and has almost the same magnitude.

3. In comparison with approximations of simple roots, the order of convergence is slow, i.e., the value of the residual is relatively large.

4. The magnitudes of the residuals for m approximations are almost the same.

After some iterations we can group the approximations into different groups by the following method. If the k-th and j-th approximations are in the same group, they must satisfy the following conditions:

1. The corrections w_k and w_j should satisfy

$$1 - \alpha < \frac{\mid w_k \mid}{\mid w_j \mid} < 1 + \alpha\,, \tag{2.16}$$

for a small number α.

2. Let θ_{kj} be the angle between vector $z_j - z_k$ and vector w_k, and θ_{jk} be the angle between vector $z_k - z_j$ and vector w_j. These angles should satisfy

$$\mid cos\theta_{kj} - cos\theta_{jk} \mid < \beta\,, \tag{2.17}$$

for a small number β. Notice that $cos\theta_{kj} = \frac{(z_j - z_k, w_k)}{|z_j - z_k||w_k|} > 0$ and $cos\theta_{jk} = \frac{(z_k - z_j, w_j)}{|z_k - z_j||w_j|} > 0$.

3. We can count the number of approximations satisfying the above two criteria to find the multiplicity. To make sure this group contains all appropriate approximations we check the criterion:

$$\mid \frac{\mid w_k^{\{r+1\}} \mid}{\mid w_k^{\{r\}} \mid} - \frac{m-1}{m+1} \mid < \gamma, \tag{2.18}$$

for a small number γ.

In our implementation, we trace the rate of convergence of each approximation by keeping track of each residual w_k. At first few iterations, w_k usually fluctuates wildly. Then all w_k start decreasing monotonically. By checking the ratio $\frac{w_k^{\{r+1\}}}{w_k^{\{r\}}}$ which converges to $(m-1)/(m+1)$ by Corollary 2.3.4, one can tell which approximations converge to simple roots. By Theorem 2.3.6 the ratios corresponding to approximations which converge to simple roots converge to zero very fast. We observed that as soon as all residuals start decreasing it takes at most 10 iterations for those approximations which converge to simple roots to converge to their targets within the chosen accuracy. Then the program begins to detect the multiplicities

and to execute the grouping process. At first, the residuals are sorted by their magnitudes and the grouping process is invoked to group the approximations according to criteria (2.16) and (2.17). The program checks criterion (2.18) for each group. If it is satisfied, the program takes the average of the approximations in the group and use it in the modified Aberth method. If criterion (2.18) is not satisfied the approximations in this group are used in the modified Aberth method instead of their average. When the grouping process is successful it takes only a few iterations for the modified Aberth method to converge since the averages are very close to the roots and the method converges cubically. In our experiment, it usually takes less than 3 iterations for modified Aberth method to converge after the grouping. In practice, the grouping process may include wrong approximations in a group. In this case, the modified Aberth method will converge linearly. The program disbands the groups with linear convergence rate and convert to the beginning with the newly computed approximations as initial approximations. The flow chart of the program is given in Figure 2.1.

2.5 A Numerical Example

We demonstrate our method by the following numerical example. Let

$$p(z) = (z - 1.1 - 1.1i)^4(z - 3.2 - 2.3i)^2(z - 2.1 - 1.5).$$

This polynomial was also used in [29] with which we will compare our result. As in [29], we choose the initial approximations $z_k = e^{2\pi ki/7}$, $k = 0, 1, \ldots, 6$. After 12 iterations the approximations and the residual $| w_i |$ are

	real	imaginary	residual
1.	2.1000000000000	1.5000000000000	0.

2.	3.2002858826098	2.2998138134465	6.8221947921266D-04
3.	3.1997141178405	2.3001862562026	6.8252015101818D-04
4.	1.0968596312555	1.0941839385686	4.2572547041395D-03
5.	1.0940624207372	1.1031845106896	4.4900998706488D-03
6.	1.1059206829313	1.0967222422851	4.5441352406740D-03
7.	1.1032928553054	1.1060241399490	4.7093618963726D-03

In fact, the 1st approximation reaches the exact root at the 10-th iteration. The 4-th, 5-th, 6-th and the 7-th are grouped as one group with multiplicity 4. Their average is

$$(1.1000338975574, 1.10002287078731).$$

It is very close to the exact root $(1.1, 1.1)$.

Grouping the 2nd and 3rd in one group gives an average

$$(3.2000000002252, 2.3000000348246)$$

which is also very close to the exact root $(3.2, 2.3)$. These demonstrate the effectiveness of Theorem 2.2.6.

Then the program switches to the modified Aberth method (2.4) . Only one iteration gives all the exact roots.

	real	imaginary
1.	1.1000000000000	1.1000000000000
2.	3.2000000000000	2.3000000000000
3.	2.1000000000000	1.5000000000000

In [29], the same groups are formed after 14 iterations. It took another 7 iterations for the program to stop. The final results for the approximations converging to the root $(1.1, 1.1)$ are listed below with accuracy to the 4-th digit.

	real	imaginary
1.	1.09914601	1.10078329
2.	1.10092914	1.09945021
3.	1.09931441	1.09917266
4.	1.10045233	1.10079471

The result is even far less accurate than our average.

In our experiments the approximations converge linearly until the modified Aberth method is applied. Once the multiplicities are detected, the modified Aberth method converges extremely fast.

Algorithm MODABERTH

```
Input: polynomial p(z) and its degree n
       the initial approximations z_i
Output: the roots of p(z)
begin
    for i = 1 to 2 do
        for j = 1 to n do
            compute new approximations z'_j
            compute residual w'_j, save old one as w_j
    for j = 1 to n do
        compute the ratio r_j = w'_j/w_j
    for i = 1 to 2
        for j = 1 to n
            compute new approximation as before
            compute residual as before
            compute ratio as before and save old ones
    !  now we have 3 consecutive ratios for each approximation
    !  we want to check if the ratios are all decreasing monotonically
    while not decreasing
        do one iteration for each approximation
    while there is an i such that r_i< small_1 and w_i > small_2
    !  if there is an approximation converging to a simple root but
    !  it has not yet converged.
        do one iteration for each approximation
    while not done
        sort residuals w_1, w_2, ..., w_n .
        for i = 1 to n and w_i > small_2
            for j = i+1 to n
                check if j should be in the same group as i
        for each group, check criterion 3
            if all groups satisfy criterion 3 set success = 1
        if success = 1
            while not all convergent
                use modified Aberth method once for each approx
                if linear convergence is observed
                    break up groups and goto begin
            done = 1 !  all approximations are convergent
        else
            use modified Aberth method once for each approx
end MODABERTH
```

Figure 2.1: **Algorithm** MODABERTH

Chapter 3

Nonsymmetric Eigenvalue Problem

3.1 Introduction

This chapter presents a parallel algorithm to solve the eigenvalues of non-symmetric matrices. Both homotopy and divide-and-conquer strategies are used. The original matrix is split into two or more submatrices, then a homotopy between the submatrices and the original matrix is established such that the intermediate matrices have only simple eigenvalues. The Aberth method is then applied to refine all the approximations simultaneously. Section 3.2 discusses the split-homotopy strategy, while Section 3.3 introduces the Hyman method for computing the characteristic polynomial and its derivative. In Section 3.4 we make a straightforward implementation on Intel iPSC/860 and Intel Delta multicomputers. However, this initial approach did not use the processors uniformly so we modify our technique in Section 3.5 to balance the load for message-passing distributed systems.

3.2 Splitting Procedure and Homotopy

Since any real matrix A can be transformed to a similar matrix in upper Hessenberg form by an orthogonal transformation, we shall assume throughout this chapter that matrix A is an upper Hessenberg matrix, namely,

$$A = (a_{ij}) = \begin{pmatrix} * & * & * & \cdots & \cdots & * \\ a_{21} & * & * & \cdots & \cdots & * \\ & a_{32} & * & \cdots & \cdots & * \\ & & \cdots & \cdots & \cdots & \cdots \\ & 0 & & a_{n-1,n-2} & * & * \\ & & & & a_{n,n-1} & * \end{pmatrix}.$$

For convenience we assume n is an even number. We further assume A is irreducible, that is, none of the subdiagonal entries $a_{j,j-1}$, $2 < j < n$, is zero, otherwise we can consider the reduced matrices. Let $k = n/2$. Let $D = (d_{i,j})$ be the matrix obtaining from A by making the subdiagonal entry $a_{k+1,k} = 0$, namely,

$$D = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}.$$

The eigenvalues of D is the union of the eigenvalues of smaller matrices A_{11} and A_{22}. It takes much less time to compute the eigenvalues of A_{11} and A_{22} and it can be done in parallel. The following homotopy is established

$$H(\lambda, t) = c(1-t)\det(D - \lambda I) + t\det(A - \lambda I) \tag{3.1}$$

where c is a random complex number and $0 \leq t \leq 1$.

For each t, since $H(\lambda, t)$ is a polynomial in λ of degree n, there are n zeros: $\lambda_1(t)$, $\lambda_2(t)$, ... , $\lambda_n(t)$. They are called the eigenpaths of system (3.1). It is well known [23] that the eigenpaths do not meet in the interval $(0, 1)$ for a randomly chosen c. It implies that the zeros of $H(\lambda, t)$ at each t, $0 < t < 1$, are distinct. Our algorithm does not depend heavily on this property, though it makes the algorithm easier to implement and more efficient.

Notice that the zeros of $H(\lambda, 0)$ are the eigenvalues of matrix D while the zeros of $H(\lambda, 1)$ are the eigenvalues of matrix A. We choose $0 = t_0 < t_1 < t_2 < \cdots < t_m = 1$

for some positive integer m. The idea is to use the zeros of $H(\lambda, t)$ at t_i as the initial approximations to the zeros of $H(\lambda, t)$ at t_{i+1}. Aberth method or our modified Aberth method may be applied to refine the approximations. It is also possible to use the tangent line at $\lambda_k(t_i)$ to predict the initial approximation for $\lambda_k(t_{i+1})$. Our numerical experiments show that the algorithm of the second approach is slower than that of the first one because of the extra calculations of derivatives in the variable t and the global convergence property of the Aberth method in practice. Here, we only discuss the first approach.

In the spirit of the Aberth method, approximations at each step must be distinct. It is possible that some of the zeros are identical at $t = 0$ that makes the Aberth method not applicable. In such case, we simply make random perturbations to make the initial approximations distinct. Notice that our goal is to find the zeros at $t = 1$. It is unnecessary to approximate the zeros with high accuracy at the intermediate steps $t_i < 1$. In our implementation, the algorithm is allowed to run for a fixed number of iterations at the intermediate steps regardless of the convergence. The global convergence property of the Aberth method in practice ensures the eventual convergence at $t = 1$.

The problem of finding the zeros at $t = 0$, i.e., the eigenvalues of both A_{11} and A_{22} are solved by the fastest available serial codes or by applying the same splitting and homotopy strategy. In the latter case, we keep dividing the matrices until certain dimension then the fastest serial code is applied. For instance, suppose the dimension of the original matrix A is $n = 2^l$, the number of processing nodes is $k = 2^m$, and $m < n$. One can split matrix A into k small matrices, all of them have dimension $L = 2^{n-m}$, the serial algorithm is then applied to those small matrices.

3.3 Hyman's Method

In order to extract the zeros of $H(\lambda,t)$ iteratively by Aberth method, $H(\lambda,t)$ and $\frac{\partial H(x,t)}{\partial \lambda}$ need to be evaluated efficiently. For this purpose, Hyman's method is used [16] which is remarkably backward stable as pointed out by Wilkinson [34]. Notice that $\det(D-\lambda I)=\det(A_{11}-\lambda I)\det(A_{22}-\lambda I)$. Differentiating equation (3.1) with respect to λ, we have,

$$\begin{aligned}\frac{\partial H(\lambda,t)}{\partial \lambda} &= c(1-t)\left[\frac{\partial \det(A_{11}-\lambda I)}{\partial \lambda}\det(A_{22}-\lambda I)+\det(A_{11}-\lambda I)\frac{\partial \det(A_{22}-\lambda I)}{\partial \lambda}\right] \\ &\quad +t\,\frac{\partial \det(A-\lambda I)}{\partial \lambda}.\end{aligned}$$

Since A_{11} and A_{22} have the same structure as A, we only need to address the problem of computing $\det(A-\lambda I)$ and $\frac{\partial \det(A-\lambda I)}{\partial \lambda}$ in more details.

For $A=(a_{ij})$ and $x=(x_1,x_2,\ldots,x_n)^T$, the system of equations $(A-\lambda I)x=0$ can be written as

$$\begin{cases}(a_{11}-\lambda)x_1+a_{12}x_2+\cdots+a_{1n}x_n=0\\ a_{21}x_1+(a_{22}-\lambda)x_2+\cdots+a_{2n}x_n=0\\ \cdots\cdots\cdots\cdots\cdots\cdots\\ a_{k,k-1}x_{k-1}+(a_{kk}-\lambda)x_k+\cdots+a_{kn}x_n=0\\ \cdots\cdots\cdots\cdots\cdots\cdots\\ a_{n,n-1}x_{n-1}+(a_{nn}-\lambda)x_n=0\,.\end{cases} \tag{3.2}$$

For given λ, letting $x_n=1$ in the last equation, we can solve the last $n-1$ equations recursively for $x_{n-1},\ldots,x_2,x_1$. These values are then used to evaluate the left-hand side of the first equation,

$$F(\lambda)=(a_{11}-\lambda)x_1+a_{12}x_2+\cdots+a_{1n}x_n\,.$$

It is clear that $F(\lambda)=0$ if λ is a eigenvalue of A.

To computer $\det(A-\lambda I)$, we write matrix $A-\lambda I$ as

$$\begin{pmatrix} a_{11}-\lambda & a_{12} & a_{13} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22}-\lambda & a_{23} & \cdots & \cdots & a_{2n} \\ & a_{32} & a_{33}-\lambda & \cdots & \cdots & a_{3n} \\ & & \vdots & \vdots & \vdots & \vdots \\ & 0 & & a_{n-1,n-2} & a_{n-1,n-1}-\lambda & a_{n-1,n} \\ & & & & a_{n,n-1} & a_{nn}-\lambda \end{pmatrix}.$$

For $j=1,2,\ldots,n$ the jth column is multiplied by x_j as found above and added to the last column, thereby yielding the matrix

$$\begin{pmatrix} a_{11}-\lambda & a_{12} & a_{13} & \cdots & \cdots & a_{1,n-1} & F(\lambda) \\ a_{21} & a_{22}-\lambda & a_{23} & \cdots & \cdots & a_{2,n-1} & 0 \\ & a_{32} & a_{33}-\lambda & \cdots & \cdots & a_{3,n-1} & 0 \\ & & \vdots & \vdots & \vdots & \vdots & \vdots \\ & 0 & & a_{n-1,n-2} & a_{n-1,n-1}-\lambda & 0 \\ & & & & a_{n,n-1} & 0 \end{pmatrix}.$$

Therefore

$$\det(A-\lambda I) = (-1)^{n-1} F(\lambda) \prod_{j=1}^{n-1} a_{j+1,j}\,.$$

To compute $\frac{\partial \det(A-\lambda I)}{\partial \lambda}$, we need to compute $\frac{\partial F}{\partial \lambda}$. Differentiating (3.2) with respect to λ, taking into account that x_j, $j=1,2,\ldots,n-1$, are functions of (λ,t), we have

$$\begin{cases} (a_{11}-\lambda)\frac{\partial x_1}{\partial \lambda} - x_1 + a_{12}\frac{\partial x_2}{\partial \lambda} + \cdots + a_{1n}\frac{\partial x_n}{\partial \lambda} = 0 \\ a_{21}\frac{\partial x_1}{\partial \lambda} + (a_{22}-\lambda)\frac{\partial x_2}{\partial \lambda} - x_2 + \cdots + a_{2n}\frac{\partial x_n}{\partial \lambda} = 0 \\ \cdots\cdots\cdots \\ a_{k,k-1}\frac{\partial x_{k-1}}{\partial \lambda} + (a_{kk}-\lambda)\frac{\partial x_k}{\partial \lambda} - x_k + \cdots + a_{kn}\frac{\partial x_k}{\partial \lambda} = 0 \\ \cdots\cdots\cdots \\ a_{n,n-1}\frac{\partial x_{n-1}}{\partial \lambda} + (a_{nn}-\lambda)\frac{\partial x_n}{\partial \lambda} - x_n = 0 \end{cases}. \tag{3.3}$$

With $\frac{\partial x_n}{\partial \lambda}=0$, we solve (3.3) successively for $\frac{\partial x_{n-1}}{\partial \lambda}, \frac{\partial x_{n-2}}{\partial \lambda}, \ldots, \frac{\partial x_1}{\partial \lambda}$, using the previously computed values of $x_1, x_2, \ldots, x_n$. The value of the left-hand side of the first equation in (3.3) is then $\frac{\partial F}{\partial \lambda}$.

3.4 Initial Implementation

We implemented the algorithm on the Intel iPSC/860 gamma multicomputer and Intel Touchstone Delta multicomputer.

Both iPSC/860 and Delta multicomputers are distributed memory MIMD machines [15]. They consist of a collection of computing nodes each of which has a processor and a local memory. The nodes are inter-connected by networks. The iPSC/860 system uses hypercube topology architecture while the Delta system uses 2 dimensional mesh. The computing nodes for both systems are the Intel *i*860 processors.

The 128 computing nodes of the iPSC/860 are networked as a 7 dimensional hypercube. The 512 computing nodes of the Delta system are networked as a 16 by 32 mesh. Each row has 32 computing nodes and each column has 16 nodes.

The communication between processors for both systems are done by a library of explicit message passing functions. It is the programmer's responsibility to locate the synchronization points of the program and specify the communication between the processors by explicitly passing messages.

It is expected that the Delta multicomputer offers better performance since for some applications 2-D mesh is a natural topology and the Delta multicomputer uses faster switching techniques [1, 6]. In our application we have not observed any significant improvement of the Delta multicomputer over the iPSC/860 system since our program doesn't need extensive communication. On the other hand, the communication in our program is solely broadcasting in which the nodes broadcast the messages in turn with no message contention. The special faster switching technique for the Delta multicomputer lose advantage for our application.

Algorithm ABERTH

```
Input: the matrix A
       the initial approximations λ_i, i = 1,2,···,n.
       the number of processors m, where n is a multiple of m
Output: the eigenvalues of A, λ_i, i = 1,2,···,n.
begin ABERTH
    s = n/m ! my share of computation load
    ! begin computation
    iam = mynode() ! system call, returns the id of this node
    for i = iam * s + 1 to (iam + 1) * s
        update λ_i ! use Aberth method
    end for
    ! end of computation
    ! begin communication
    for i = 0 to m − 1 ! the node counts from 0 to m − 1
        if (iam = i) then
            send λ_{iam*s+1},···,λ_{(iam+1)*s} to all other nodes.
        else
            receive λ_{iam*s+1},···,λ_{(iam+1)*s} from node i.
        end if
    end for
    ! end communication
end ABERTH
```

Figure 3.1: **Algorithm** ABERTH

Both systems support Fortran 77 with message passing libraries. Their main differences are their inter-connection topologies and the switching techniques which are invisible to the users. The programs written for one system are portable to the other. In our implementation, programs were written for iPSC/860 multicomputer first, then ported to the Delta multicomputer without modification.

The program has two phases. The first phase splits the matrix into two or more submatrices. Then subroutine HQR in EISPACK is called to find the eigenvalues of each submatrices. Those eigenvalues are used in the second phase as the initial approximations to the eigenvalues of the target matrix. The Aberth method is then

used iteratively to find all the eigenvalues in parallel. The basic structure of the program for one iteration in the second phase is shown in Figure 3.1.

When compared to the fastest available sequential code *HQR* in EISPACK running on one node, our parallel program was slower when less than eight nodes were used, see Table 3.1, Table 3.2 and Table 3.3 for matrices with different dimensions. For eight or more nodes, the parallel version was faster, by a factor of two on 16 nodes. Execution time improved as the number of nodes increased. As more nodes are added the improvement flattens off due to the load imbalancing problem and the communication overhead.

	nodes	sec
HQR on 1 node 0.75 sec	1	2.72
	2	1.55
	4	0.97
	8	0.67
	16	0.52
	32	0.47
	64	0.60

Table 3.1: Time on i860 with a 64×64 random matrix.

	nodes	sec
HQR on 1 node 4.56 sec	1	18.96
	2	10.61
	4	6.63
	8	4.23
	16	2.83
	32	2.25
	64	2.48
	128	2.72

Table 3.2: Time on i860 with a 128×128 random matrix.

	nodes	sec
HQR on 1 node 32.24 sec	1	149.45
	2	84.54
	4	51.22
	8	30.29
	16	17.87
	32	10.81
	64	8.21
	128	9.82
	256	13.38

Table 3.3: Time on i860 with a 256×256 random matrix.

3.5 Load-Balancing

In this section we consider the load-balancing problem. The Aberth method fits parallel computers naturally. Suppose n nodes are used for the computation of the n eigenvalues of an $n \times n$ matrix. Each node is responsible for finding one eigenvalue. For direct implementation, every node has a copy of the complete information of the matrix and all the n initial approximations. The nodes execute calculations simultaneously so that one node updates only one approximation. Since all nodes have the same amount of computation, they generally complete the calculation at the same time. Then every node broadcasts the updated approximation to the other nodes. This process continues until all the nodes find the expected eigenvalues. A natural stopping criterion is, as soon as an approximation is good enough no further iteration on that approximation is needed.

In Section 3.4 we have shown that the parallel algorithm can beat the best serial algorithm, *HQR* in EISPACK, for a sufficiently large number of processors using an Intel Delta multicomputer. However, if the number of nodes is far less than the number of eigenvalues, each node is responsible for finding more than one eigenvalue and

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16
N0	10	10	10	10	9	7	7	5	5	5	3	2	1	1	1	0
N1	10	10	9	8	6	4	3	3	2	2	2	1	1	1	0	0
N2	10	10	10	10	9	9	8	8	8	7	7	6	3	2	1	1
N3	10	10	10	9	9	8	7	5	5	4	3	3	2	1	0	0
N4	10	10	10	9	9	8	7	7	6	6	6	4	4	3	1	1

Table 3.4: Unbalanced Load Distribution

some nodes may complete their jobs earlier. This difference causes load-imbalance.

To illustrate the load-imbalance, let's compute the eigenvalues of a 50×50 matrix whose eigenvalues and the initial approximations are randomly chosen in the region $\{z \in C, |z| < 1\}$. With 5 nodes, the program assigns the first 10 eigenvalues to the first node, the second 10 eigenvalues to the second node, ... , and the last 10 eigenvalues to the last node. Since it takes the same number of floating-point operations for any node to update one approximation, we will count the amount of time to complete this number of operations as one unit time. Table 3.4 lists the number of eigenvalues each node computes at each step—each row represents a node and each column represents a step. It takes 105 parallel time units for the program to complete.

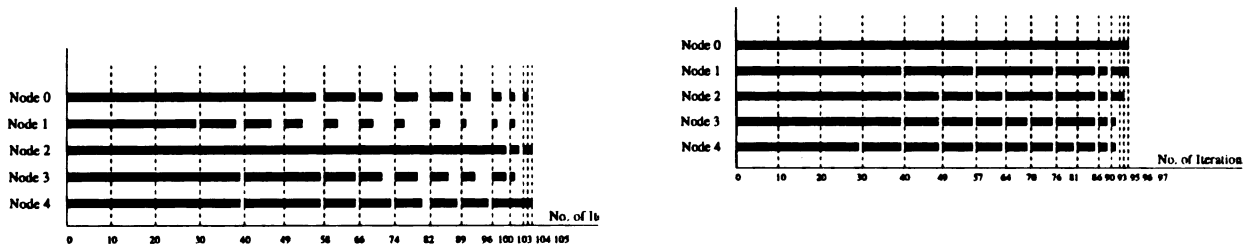
The key for the load-balancing problem for the Aberth method is the observation that if every node has a complete copy of the previous approximations, it can update any approximation. The total number of eigenvalues is sufficiently small for each node to have a copy. Based on the differences between the old approximations and the new ones it received, each node can decide which approximations are good enough and need no more updating. Once a node has complete information on which approximations need to be updated and the number of nodes involved in the computation, it can determine the distribution of approximations to nodes. The number of approx-

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16
N0	10	10	10	10	9	8	7	6	6	5	5	4	3	2	1	1
N1	10	10	10	9	9	7	7	6	5	5	4	3	2	2	1	1
N2	10	10	10	9	8	7	6	6	5	5	4	3	2	2	1	0
N3	10	10	10	9	8	7	6	5	5	5	4	3	2	1	0	0
N4	10	10	9	9	8	7	6	5	5	4	4	3	2	1	0	0

Table 3.5: Balanced Load Distribution

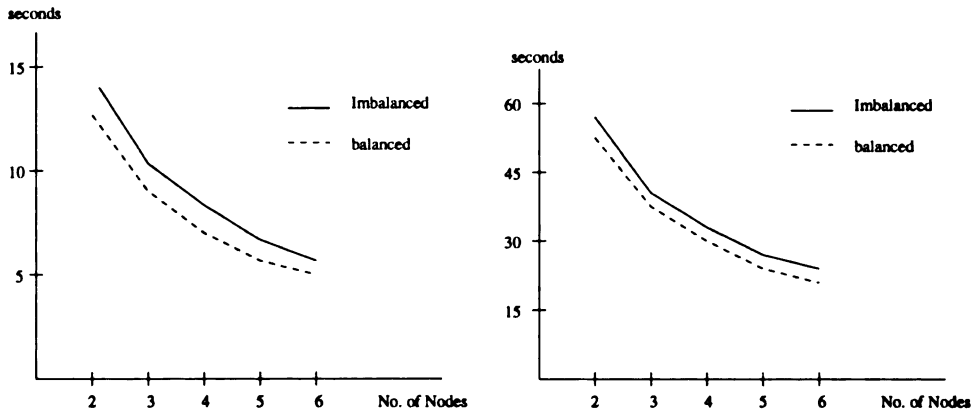
imations can be evenly divided among nodes, and that determination is completely distributed. Note that each node also knows the approximations other nodes are to update. This information is required when a node broadcasts the updated approximations later. Node i updates $\lfloor \frac{m}{nnodes} \rfloor + r_i$ consecutive approximations, where m is the number of approximations to be updated at this step, *nnodes* is the number of nodes involved in the computation, $r_i = 1$ if $i < m - \lfloor \frac{m}{nnodes} \rfloor$ and $r_i = 0$ otherwise. For example, when there are 50 approximations to be updated and there are 5 nodes, each node gets 10 consecutive approximations. If there are 33 approximations, then node 0, node 1 and node 2 each gets 7 consecutive approximations while node 3 and node 4 each gets 2 consecutive approximations. For this balanced program, at each step every node computes the approximations it is responsible for. Table 3.5 lists the number of approximations each node computes at each step for the above example. The program takes 97 parallel time units to complete which represents a distinct improvement over the 105 time units for the unbalanced example. Figure 3.2 shows the corresponding timing information.

The unbalanced program uses static scheduling which assigns the i-th chunk of approximations to the i-th node. The balanced program, on the other hand, assigns the i-th share of approximations to the i-th node dynamically at each step. For this



(a) Unbalanced Timing (b) Balanced Timing

Figure 3.2: Unbalanced vs. Balanced Timing



(a) Timing for 240 × 240 matrices (b) Timing for 360 × 360 matrices

Figure 3.3: Timing Info for 240 × 240 and 360 × 360 matrix

dynamical scheduling extra computation is needed to determine the shares for all the nodes at every node. Fortunately, this computation takes only $O(n)$ operations, while one updating of an approximation takes $O(n^2)$ operations in Hyman's method to evaluate the polynomial and its derivative. This extra computation is negligible.

We implemented the balanced algorithm on a cluster of up to 6 DEC Alfa 3000 workstations running Message-Passing-Interface (MPI) with Fortran 77. MPI is a widely used standard for writing message-passing programs. It is available for almost all existing high performance machines [27]. The test matrices are real upper Hessenberg matrices whose entries are randomly generated in the range $(-10, 10)$.

The matrix is divided in the middle into two smaller real upper Hessenberg matrices. Subroutine HQR in EISPACK is called to find the eigenvalues for each of the two matrices. The eigenvalues for the two matrices are merged as the initial approximations to the original matrix. Aberth method is applied to refine the approximations. We tested the matrices of degree up to 360×360. The results show that the balanced program saves about 10% for the chosen class of matrices compared to the unbalanced program. Figures 3.3 shows the timing information for test matrices of degree 240×240 and degree 360×360.

3.6 Conclusion

In this chapter we describe a parallel algorithm to solve the eigenvalues of non-symmetric matrices. The straightforward approach leads to an unbalanced utilization of processors so a dynamically load balanced approach was developed and demonstrated. At this time, only six processors were available so the gains are small. Since the computations grow much faster than communication we expect significantly better results on a larger machine, especially with respect to the fastest serial algorithm, HQR.

Bibliography

[1] Aad J. wan der Steen, *An overview of (almost) available parallel systems*, Publication of the NCF, December 1993.

[2] O. Aberth, *Iteration Methods for Finding all Zeros of a Polynomial Simultaneously*, Math. Comp., Vol.27,1973, pp. 339-344

[3] G. Alefeld and Herzberger, *On the Convergence speed of Some algorithms for simultaneous approximation of polynomial roots*, SIAM J. Num. Anal., 11(1974), 237-243.

[4] Alan F. Beardon, *Iteration of Rational Functions*, Springer-Verlag, 1990.

[5] W. Borsch-Supan, *A posteriori error bounds fro the zeros of polynomials*, Numer. Math. 6(1963) pp. 380-398.

[6] Lionel M. Ni, *Lecture Notes of CPS822*, Computer Science Department, Michigan State University, Spring, 1995.

[7] J.Cuppen, *A divide and comquer method for the symmetric tridiagonal matrices*, Numer. Math., 36(1981), pp. 177-195.

[8] K. Docev, *A modified Newton method for the simultaneous approximation of all roots of the given algebraic equations*, Fiz. - Mat. Spis Bulgar. Akad. Nauk 5 (1962) pp. 136-139 (in Bulgarian).

[9] J. Dongarra, M. Sidani, *A parallel algorithm for the nonsymmetric eigenvalue problem*, SIAM J. Sci. Comput. 5(1993), pp. 542-569.

[10] J. Dongarra, C. Sorensen, *A fully parallel algorithm for the symmetric eigenvalue problem*, SIAM J. Sci. Statist. Comput. 8(1978), pp. S139-@154.

[11] E. Durand, *Solutions Numeriques des Equations Algebriques*, T.I. Paris 1960.

[12] L.W. Ehrlich, *A modified Newton method for polynomials*, , Comm. ACM 10 (1967) pp. 107-108.

[13] M.R.Farmer,G. Loizou, *A CLASS OF ITERATION FUNCTIONS FOR IMPROVING,SIMULTANEOUSLY, APPROXIMATIONS TO THE ZEROS OF A POLYNOMIAL*, BIT 15 (1975),pp. 250 - 158.

[14] Hamilton, *A Type of Variation on Newton's Method*, Amer. Math. Monthly, 57, 517-522(1950).

[15] Kai Hwang, *Advanced Computer Architecture: Parallelism, Scalability, Programmability*, McGraw-Hill Inc. 1993.

[16] M. Hyman, *Eigenvalues and Eigenvectors of general matrices*, presented at the 12th National Meeting of the Association for Computing Machinary, Houston, Texas, June 1957.

[17] Pierre Fraigniaud, *The Durand-Kerner Polynomials Roots-Finding Method in case of Multiple Roots*, BIT 31(1991),112-123.

[18] T.L.Freeman, *Calculating Polynomial zeros on a local memory parallel computer*, Paralle Computing 12(1989) 351-358.

[19] M.W. Green, A.J.Korsak and M.C.Pease, *Simultaneous iteration towards all roots of a complex polynomial*, SIAM Rev. 18 (1976), pp. 501-502.

[20] I Kerner, *Ein Gessamtschrittverfahren zur Berechung der Nullstellen von Polynooem.* Num. Math. 8(1966), 290-294.

[21] Göran Kjellberg, *Two Observations on Durand-Kerner's Root-Finding Method*, BIT 24(1984),556-559.

[22] K. Li and T.Y. Li, *An Algorithm for Symmetric Tridiagonal Eigenproblems–Divide and Conquer with Homotopy Continuation*, SIAM J. Sci. Statist. Comput., 14(1993), pp. 735-751.

[23] T.Y. Li, T. Sauer, J. Yorke, *The random product homotopy and deficient polynomial systems*, Numer. Math., 51, (1987), pp. 481-500.

[24] T.Y. Li , Zhonggang Zeng, *Homotopy-determinant algorithm for solving nonsymmetric eigenvalue problems*, Math. Comp. 10(1992), pp. 483-502.

[25] T.Y. Li and Zhonggang Zeng, *The Laguerre Iteration in Solving the Symmetric Tridiagonal Eigenproblem, Revisited*, SIAM J. Sci. Comput. 9(1994), 1145-1173.

[26] T.Y.Li, Z. Zeng, and L. Cong, *Solving eigenvalue Problems of real nonsymmetric matrices with real homotopies*, SIAM J. Numer. Anal. 29(1992), 229-248.

[27] Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard*, May 5, 1994.

[28] M. Marden, *Geometry of Polynomials*, Amer. Mathematical Soc., Providence, RI, 2nd ed., 1966.

[29] Tsuyako MIYAKODA, *Iterative method for multiple zeros of a polynomial by clustering*, J. Com. and App. Math. 28(1989),pp. 315-326.

[30] J.M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, 1970.

[31] B.N.Parlett, *Laguerre's Method Applied to Matrix Eigenvalue Problem*, Math. Comp. 18(1965), pp. 464-485.

[32] L. Pasquini and D. Trigiante, *Il metodo di continuazione e l'approssimazione simultanea degli zeri dk un polinomio*, Monografie di Soft. Matem., N. 30, Pubbl. dell'IAC, 1984.

[33] T. Terano, *On a global algorithm for algebraic equationss*, PhD Thesis, University of Tokyo, Information engineering course (1978).

[34] J.H. Wilkinson, *Error Analysis of Floating-point Compuation*, Numer. Math. 2 (1960), 319-340.