Contents

Chapter 6

Solving Scalar Nonlinear Equations

6.1 Some Basic Concepts and Methods

6.1.1 Introduction

In this chapter we study numerical methods for computing accurate approximations to the roots of a scalar nonlinear equation

$$f(x) = 0, \tag{6.1.1}$$

where $f(x)$ is a real-valued function of one variable. This problem has occupied mathematicians for many centuries and many of the basic methods date back a long time. In general the roots of (6.1.1) cannot be expressed in closed form. Even when an explicit solution is available (as, e.g., for the reduced cubic equation), this is often so complicated that using, e.g., Newton's method, is more practical; see Problem 2.3.8.

Numerical methods are iterative in nature. Starting from one or more initial approximations, they produce a sequence of approximations, which presumably converges to the desired root. Note that the function $f(x)$ need not be known by a closed analytical expression. For numerical methods to be applicable it suffices that $f(x)$, and preferably some of its derivatives, can be evaluated for given numerical values of x. It is not uncommon with applications where each function value is obtained by a complicated computation, e.g., by the numerical solution of a differential equation. The object is then to use as few function evaluations as possible in order to approximate the root with a prescribed accuracy.

Iterative methods have to be truncated after a finite number of steps and therefore can yield only approximations to the desired roots. Further, the roundoff errors that occur in the evaluation of $f(x)$ will limit the accuracy attainable by *any numerical method*. The effect of such rounding errors depends on the conditioning of the roots and is discussed in Section 6.1.3.

With certain methods it is sufficient for convergence to know an initial interval $[a, b]$, which contains the desired root (and no other root). An important example is the bisection method described in Section 6.1.2. It is often suitable to use a hybrid method in which the bisection method is used to roughly locate the root. A more rapidly convergent method is then used to refine this approximation. These latter methods make more use of regularity assumptions about $f(x)$, and usually also require an initial approximation close to the desired root.

The theory of fixed point iteration methods is treated in Sec. 6.1.4 and the concepts of convergence order and efficiency introduced in Sec. 6.1.5. The secant method, and other methods based on interpolation are described in Sec. 6.2. In Sec. 6.4 we briefly consider methods for solving the related problem of finding the minimum or maximum of a real-valued function $g(x)$. Newton's method and other methods of higher order are analyzed in Sections 6.3. A classical problem is that of determining all real or complex roots of a algebraic equation. Special features and methods for this problem are taken up in Section 6.5.

Many of the methods for a single equation, such as Newton's method, are easily generalized for *systems of nonlinear equations*. However, unless good approximations to the roots are known, several modifications of the basic methods are required, see Vol. II, Chapter 11.

6.1.2 The Bisection Method

It is often advisable to start with collecting some qualitative information about the roots to be computed. One should try to determine how many roots there are and their approximate location. Such information can often be obtained by graphing the function $f(x)$. This can be a useful tool for determining the number of roots and intervals containing each root.

Example 6.1.1.

Consider the equation

$$f(x) = (x/2)^2 - \sin x = 0.$$

In Figure 6.1.2 the graphs of $y = (x/2)^2$ and $y = \sin x$ are shown. Observing the intersection of these we find that the unique positive root lies in the interval $(1.8, 2)$, probably close to $\alpha \approx x_0 = 1.9$.

The following **intermediate-value theorem** can be used to infer that an interval $[a, b]$ contains *at least one root* of $f(x) = 0$.

Theorem 6.1.1.

Assume that the function $f(x)$ is continuous for $a \leq x \leq b$, $f(a) \neq f(b)$, and k is between $f(a)$ and $f(b)$. Then there is a point $\xi \in (a, b)$, such that $f(\xi) = k$. In particular, if $f(a)f(b) < 0$ then the equation $f(x) = 0$ has at least one root in the interval (a, b).

A systematic use of the intermediate-value theorem is made in the **bisection method**. Assume that $f(x)$ is continuous in the interval (a_0, b_0) and that

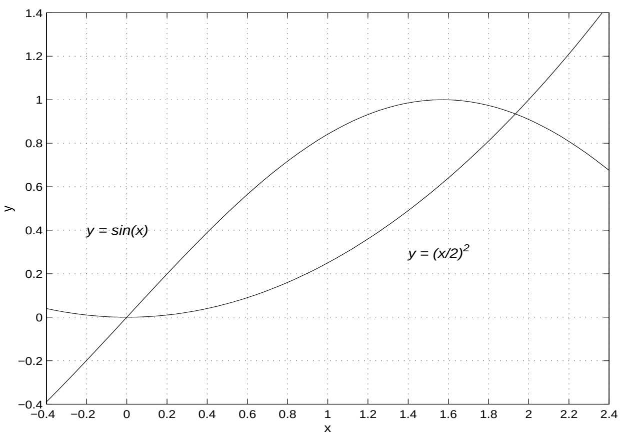

Figure 6.1.1. *Graph of curve $y = (x/2)^2 - \sin x$.*

$f(a_0)f(b_0) < 0$. We shall determine a nested sequence of intervals $I_k = (a_k, b_k)$, $k = 1, 2, 3, \ldots$, such that

$$(a_0, b_0) \supset (a_1, b_1) \supset (a_2, b_2) \subset \cdots$$

and which all contain a root of the equation $f(x) = 0$. The intervals are determined recursively as follows. Given $I_k = (a_k, b_k)$ compute the midpoint

$$m_k = \frac{1}{2}(a_k + b_k) = a_k + \tfrac{1}{2}(b_k - a_k).. \tag{6.1.2}$$

and $f(m_k)$. The latter expression has the advantage that using this to compute the midpoint no rounding error occurs in the subtraction (see Theorem 2.2.2).

We can assume that $f(m_k) \neq 0$, since otherwise we have found a root. The new interval $I_{k+1} = (a_{k+1}, b_{k+1})$ is then determined by the rule

$$(a_{k+1}, b_{k+1}) = \begin{cases} (m_k, b_k), & \text{if } f(m_k)f(a_k) > 0; \\ (a_k, m_k), & \text{if } f(m_k)f(a_k) < 0. \end{cases} \tag{6.1.3}$$

From the construction it follows immediately that $f(a_{k+1})f(b_{k+1}) < 0$ (see also Figure 6.1.2) and therefore the interval I_{k+1} also contains a root of $f(x) = 0$.

After n bisection steps we have contained a root in the interval (a_n, b_n) of length $2^{-n}(b_0 - a_0)$. If we take m_n as an estimate of the root α, we have the error estimate

$$|\alpha - m_n| < 2^{-(n+1)}(b_0 - a_0). \tag{6.1.4}$$

At each step we gain one binary digit in accuracy or, since $10^{-1} \approx 2^{-3.3}$, on the average one decimal digit per 3.3 steps. To find an interval of length δ which includes a root will require about $\log_2((b - a)/\delta)$ evaluations of f. Note that the bisection algorithm makes no quantitative use of the magnitude of computed function values.

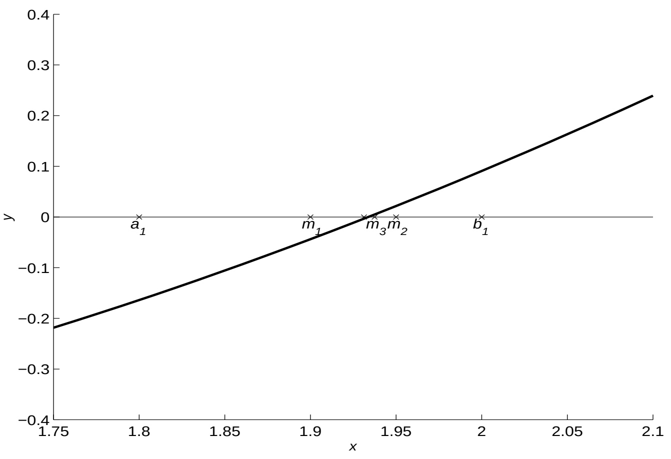

Figure 6.1.2. *The bisection method.*

Example 6.1.2.

The bisection method applied to the equation $(x/2)^2 - \sin x = 0$, with $I_0 = (1.8, 2)$ gives the sequence of intervals $[a_n, b_n]$, *where*:

k	a_k	b_k	m_k	$f(m_k)$
1	1.8	2	1.9	<0
2	1.9	2	1.95	>0
3	1.9	1.95	1.925	<0
4	1.925	1.95	1.9375	>0
5	1.925	1.9375	1.93125	<0
6	1.93125	1.9375	1.934375	>0

Here after six function evaluations we have $\alpha \in (1.93125, 1.934375)$ an interval of length $0.2 \cdot 2^{-6} = 0.003125$.

Example 6.1.3.

The inequalities $a \leq \frac{1}{2}(a + b) \leq b$, where a and b are floating point numbers with $a \leq b$ can be violated in base 10 arithmetic. For example, assume that floating point arithmetic with six decimal digits is used. Taking $a = 0.742531$ and $b = 0.742533$ we obtain $fl(a+b) = 1.48506$ (rounded) and $\frac{1}{2}(a+b) = 0.742530$. On the other hand the inequalities $a \leq a + \frac{1}{2}(b - a) \leq b$ are true in base β arithmetic, for any β. With a and p as given we get the correct value 0.742532.

An algorithmic description of the bisection method is given below. In this the tolerance τ is increased by the amount $u \max(|a|, |b|)$, where u is the machine precision. This is to guard against the possibility that δ has been chosen smaller than the spacing between the floating point numbers between a and b.

Algorithm 6.1.1 *The Bisection Method.*

Let f be a given function and $I = [a, b]$ an interval such that $b > a$ and $f(a)f(b) \le 0$. The algorithm bisect attempts to compute an approximation to a root $m \in I$ of $f(x) = 0$, with an error less than a specified tolerance $\tau > 0$.

```
function  $r = \text{bisect}(f, a, b, \tau)$;
$fa = f(a)$;
$fb = f(b)$;
while $|b - a| > \tau + u \cdot \max((|a|, |b|)$;
    $m = a + (b - a)/2$;
    $fm = f(m)$;
    if $fm \cdot fa \le 0$
      $b = m$;   $fb = fm$;
    else
      $a = m$;   $fa = fm$;
    end;
end;
$r = a + (b - a)/2$;
```

The time required by the bisection algorithm is typically proportional to the number of function evaluations, other arithmetic operations being insignificant. The correct subinterval will be chosen in the algorithm as long as the sign of the computed function value $f(m)$ is correctly determined. If the tolerance τ is taken too "small" or the root is ill-conditioned this may fail to be true in the later steps. Even then the computed midpoints will stay within a certain domain of uncertainty. Due to rounding errors there is a limiting accuracy, with which a root can be determined from approximate function values; see in Section 6.1.3.

The bisection method is optimal for the class of functions that changes sign on $[a, b]$ in the sense that it minimizes the maximum number of steps over all such functions. The convergence is rather slow, but *independent of the regularity of* $f(x)$. For other classes of functions, e.g., functions that continuously differentiable on $[a, b]$, methods like Newton's method, which assume some regularity of $f(x)$ can achieve significantly faster convergence.

If $f(a)f(b) < 0$ then by the intermediate value theorem the interval (a, b) contains *at least one root* of $f(x) = 0$. If the interval (a, b) contains several roots of $f(x) = 0$, then the bisection method will converge to just one of these. (Note that there may be one or several roots in (a, b), also in case $f(a)f(b) > 0$.)

If we only know (say) a lower bound $a < \alpha$ for the root to be determined we can proceed as follows. We choose an initial steplength d and in the first **hunting phase** compute successively function values $f(a + h)$, $f(a + 2h)$, $f(a + 4h), \ldots$, i.e. we double the step, until a function value is found such that $f(a)f(a + 2^k h) < 0$. At this point we have bracketed a root and can initiate the bisection algorithm.

In the bisection method the interval of interest is in each step split into *two* subintervals. An obvious generalization is to partition instead into k subintervals, for $p \geq 2$. In such a **multi-section method** of order p the interval $I = [a, b]$ is divided into k subintervals $I_i = [x_i, x_{i+1}]$, where

$$x_i = a + i\,[(b-a)/p], \quad i = 0 : p.$$

If there exists only one root in the interval I and we wish to compute it with an absolute error ϵ, then it is necessary to perform

$$n_k = \log_2\left(\frac{b-a}{2\epsilon}\right) \Big/ \log_2(p)$$

multi-sections of order p. Thus, the efficiency of multi-section of order p compared to bisection ($p = 2$) is

$$n_2/(pn_p) = \log_2(p)/p.$$

Hence if there is a single root in the interval bisection is always preferable. If there are several roots in the interval multi-section may perform better if the subintervals can be processed in parallel.

There are several other applications of the bisection algorithm. For example, in Section 4.4.5 we considered evaluating the nonzero B-splines for a given argument x. Then we first have to search an ordered sequence of knots $\tau_0, \dots, \tau_m$ to find the interval such that $\tau_j \leq x < \tau_{j+1}$. This can be achieved by a slight modification of the bisection method. A similar problem, important in computer science, is *searching in an ordered register*, e.g., a register of employees ordered according to increasing Social Security number. If the nth number in the register is denoted by $f(n)$, then searching for a certain number a means that an equation $f(n) = a$ is to be solved (here f is an increasing, discontinuous function). The bisection method can also be used in searching an *alphabetically* ordered register.

In later sections we will study methods for solving a nonlinear equation, which make more efficient use of computed function values than the bisection method and possibly also use values of derivatives of $f(x)$. If $f(x)$ is sufficiently regular such methods can achieve significantly faster convergence.

6.1.3 Attainable Accuracy and Termination Criteria

In the following we denote by $\overline{f}(x) = fl(f(x))$ the limited-precision approximation obtained when $f(x)$ is evaluated in floating point arithmetic. When a monotone function $f(x)$ is evaluated in floating point arithmetic the resulting approximation $\overline{f}(x)$ is not in general monotone. The effect of rounding errors in evaluating a certain polynomial of fifth degree with a simple zero at $x = 1$ is illustrated in Figure 6.1.4. Note the loss of monotonicity caused by rounding errors. This figure also shows that even if $\overline{f}(a)\overline{f}(b) < 0$, the true equation $f(x) = 0$ may not have a zero in $[a, b]$!

Even if the true function value $|f(x_n)|$ is "small" one cannot deduce that x_n is close to a zero of $f(x)$ without some assumption about the size of the derivative of f. We recall some basic results from analysis; for proofs see, e.g., Ostrowski [19, Chapter 2].

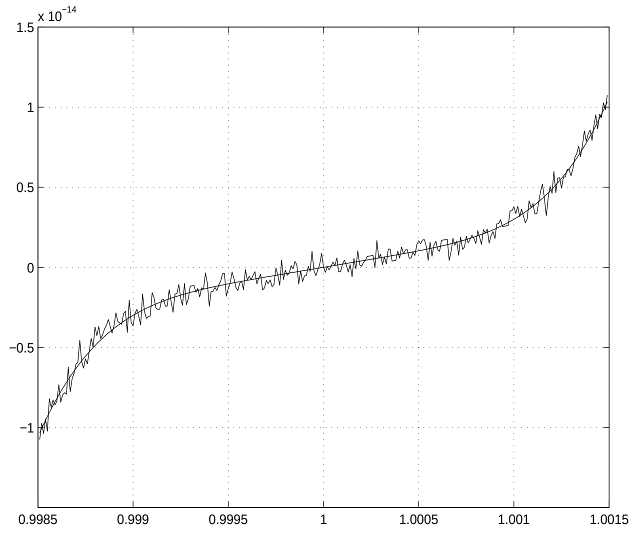

Figure 6.1.3. *Limited-precision approximation of a continuous function.*

Theorem 6.1.2.

Let $f(x)$ be continuous and differentiable in the interval $J = [x_n - \eta, x_n + \eta]$ for some $\eta > 0$. If $|f'(x)| \geq m_1$ for all $x \in J$ and $|f(x_n)| \leq \eta m_1$ then $f(x)$ has exactly one zero in J.

A root α of $f(x) = 0$ is said to be **simple** root if $f'(\alpha) \neq 0$. We now derive an error estimate for a simple root α of $f(x)$, which takes into account errors in the computed values of $f(x)$. Assume that

$$\overline{f}(x) = f(x) + \delta(x), \quad |\delta(x)| \leq \delta, \quad x \in J, \tag{6.1.5}$$

where δ is an upper bound for rounding and other errors in computed function values of $f(x)$. Using Theorem 6.1.2 we obtain

$$|x_n - \alpha| \leq \eta = (|\overline{f}(x_n)| + \delta)/m_1, \quad |f'(x)| \geq m_1, \quad x \in J. \tag{6.1.6}$$

Obviously the best we can hope for is to find an approximation x_n such that the computed function value $\overline{f}(x_n) = 0$. It follows that *for any numerical method*, δ/m_1 is an approximate limit for the accuracy with which a simple zero α can be determined. If $f'(x)$ does not vary much near $x_n = \alpha$, then we have the approximate error bound

$$|x_n - \alpha| \leq \delta/m_1 \approx \epsilon_\alpha, \quad \epsilon_\alpha = \delta/|f'(\alpha)|. \tag{6.1.7}$$

Since this is the best error bound for *any* method, we call ϵ_α, the **attainable accuracy** for the simple root α, and the interval $[\alpha - \epsilon_\alpha, \alpha + \epsilon_\alpha]$ the **domain of uncertainty** for the root α. If $|f'(\alpha)|$ is small, then ϵ_α is large and the problem of computing the root α is ill-conditioned (see again Figure 6.1.3).

Example 6.1.4.

Suppose we have computed the approximation $x = 1.93375$ to the positive root to the equation $f(x) = \sin x - (x/2)^2$. We have $f'(x) = \cos x - x/2$ and it is easily verified that $|f(x)| > 1.31 = m_1$, $x \in [1.93, 1.94]$. Further, using six decimals we have $\sin 1.93375 = 0.934852 \pm 0.510^{-6}$, and $(x/2)^2 = 0.966875)^2 = 0.934847 \pm 0.510^{-6}$. Then from (6.1.6) follows the strict error estimate

$$|x - \alpha| < 6 \cdot 10^{-6}/1.31 < 5.6 \cdot 10^{-6}.$$

Using the following theorem, an analogous result can be shown for zeros of a complex function $f(z)$ of a complex variable z.

Theorem 6.1.3.

Let $f(z)$ be analytic in the disc $K = \{z \mid |z - z_0| \le \eta\}$ for some $\eta > 0$. If $|f'(z)| \ge m$ in K and $|f(z_0)| \le \eta m$ then $f(z)$ has a zero inside K.

The **multiplicity** of a root is defined as follows:

Definition 6.1.4.

Suppose that $f(x)$ is q times continuously differentiable in a neighborhood of a root α to the equation $f(x) = 0$. Then α is said to have multiplicity q if

$$0 \ne \lim_{x \to \alpha} |f(x)/(x - \alpha)^q| < \infty. \tag{6.1.8}$$

If a root α has multiplicity q then by (6.1.8) $f^{(j)}(\alpha) = 0$, $j < q$ and from Taylor's formula

$$f(x) = \frac{1}{q!}(x - \alpha)^q f^{(q)}(\xi), \quad \xi \in \operatorname{int}(x, \alpha). \tag{6.1.9}$$

Assuming that $|f^{(q)}(x)| \ge m_q$, $x \in J$, and proceeding as before, we find that the attainable accuracy for a root of multiplicity q is given by

$$|x_n - \alpha| \le (q!\,\delta/m_q)^{1/q} \approx \epsilon_\alpha, \quad \epsilon_\alpha = (q!\,\delta/|f^{(q)}(\alpha)|)^{1/q}. \tag{6.1.10}$$

Comparing this with (6.1.7), we see that because of the exponent $1/q$ multiple roots are in general very ill-conditioned. A similar behavior can be expected also when there are several distinct but “close” roots. An instructive example is the Wilkinson polynomial, studied in Example 6.5.1.

Example 6.1.5.

The equation $f(x) = (x - 2)x + 1 = 0$ has a double root $x = 1$. The (exact) value of the function at $x = 1 + \epsilon$ is

$$f(x + \epsilon) = (\epsilon - 1)(1 + \epsilon) + 1 = -(1 - \epsilon^2) + 1 = \epsilon^2.$$

Now, suppose that we use a floating point arithmetic with eight decimal digits in the mantissa. Then

$$fl(1-\epsilon^2) = 1, \quad |\epsilon| < \frac{1}{2}\sqrt{2}\cdot 10^{-4},$$

and for $0.99992929 \leq x \leq 1.0000707$, the computed value of $f(x)$ will be zero when $f(x)$ is evaluated using Horner's rule. Hence the root can only be computed with about four correct digits, that is, with a relative error equal to the *square root of the machine precision.*

Suppose that we want to compute an approximation to a simple root α to a prescribed accuracy. Provided that the absolute value of the derivative is easy to estimate it may be possible to interrupt the iterations on the basis of the error estimate (6.1.7). However, on a computer it is usually more effective to iterate a few extra times, rather than make the effort to use a special formula for error estimation.

In subroutines for solving a nonlinear equation it is common practice to use a termination criterion of the following form. Assuming the method produces a seqeunce of bracketing intervals $[a_k, b_k]$ the iterations are terminated if

$$|b_k - a_k| \leq 2u|x_n| + \tau, \tag{6.1.11}$$

where τ is a user specified absolute tolerance and u is the rounding unit (see Section 2.2).

We must also deal with the possibility that the user specified tolerance is too small and cannot be attained. If this is the case, then from some step onwards rounding errors will dominate in the evaluation of $f(x_n)$ and the computed values of $f(x)$ may vary quasi-randomly in the interval $(-\delta, \delta)$ of attainable accuracy. If we are using a method like the bisection method, the iterations will continue until the criterion (6.1.11) is satisfied, but this, of course, does *not* ensure that the root actually has been determined to this precision!

The following alternative termination criterion can be used for superlinearly convergent methods:[1] *Accept the approximation x_n when for the first time the following two conditions are satisfied:*

$$|x_{n+1} - x_n| \geq |x_n - x_{n-1}|, \qquad |x_n - x_{n-1}| < tol. \tag{6.1.12}$$

Here *tol* is a coarse tolerance, used only to prevent the iterations from being terminated before x_n even has come close to α. When (6.1.12) is satisfied the attainable accuracy has been reached and the quantity $|x_{n+1} - x_n|$ usually is a good estimate of the error $|x_n - \alpha|$. Using this criterion the risk of never terminating the iterations for an ill-conditioned root is quite small. Note also that iteration methods of superlinear convergence ultimately converge so fast that the cost of always iterating until the attainable accuracy is obtained may be small, even if the user specified tolerance is much larger than ϵ_α.

[1]This criterion was suggested by the Norwegian computer scientist Jan Garwick.

6.1.4 Fixed-Point Iteration

We now introduce a very general class of iteration methods, which includes many important root finding methods as special cases.

Let ϕ be a continuous function and $\{x_n\}$ the sequence generated by

$$x_{n+1} = \phi(x_n), \quad n = 0, 1, 2, \dots . \tag{6.1.13}$$

for some initial value x_0. Assuming that $\lim_{n\to\infty} x_n = \alpha$, it follows that

$$\alpha = \lim_{n\to\infty} x_n = \lim_{n\to\infty} \phi(x_n) = \phi(\alpha), \tag{6.1.14}$$

i.e., the limiting value α is a root of the equation $x = \phi(x)$. We call α a **fixed point** of the mapping $x \to \phi(x)$ and the iteration (6.1.13) a **fixed point iteration**.

An iterative method for solving an equation $f(x) = 0$ can be constructed by rewriting it in the equivalent form $x = \phi(x)$, which then defines a fixed point iteration (6.1.13). Clearly this can be done in many ways. For example, let $g(x)$ be any function such that $g(\alpha) \neq 0$ and set

$$\phi(x) = x - f(x)g(x). \tag{6.1.15}$$

Then α is a solution to $f(x) = 0$ if, and only if, α is a fixed point of ϕ

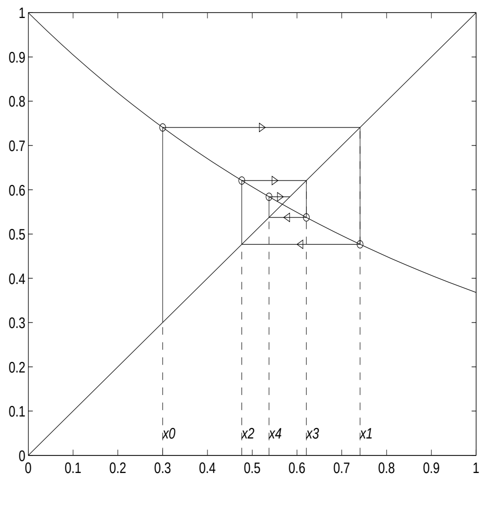

Figure 6.1.4. *The fixed point iteration* $x_{k+1} = e^{-x_k}$, $x_0 = 0.3$.

Example 6.1.6.

The equation $x + \ln x = 0$ can, for example, be written as:

(i) $x = -\ln x$; (ii) $x = e^{-x}$; (iii) $x = (x + e^{-x})/2$.

Each of these give rise to a different fixed point iteration. Results from the first eight iterations

$$x_{n+1} = e^{-x_n}, \quad x_0 = 0.3,$$

are pictured in Figure 6.2.1. The convergence is slow and we get $x_9 = 0.5641$ (correct value 0.567143).

As was shown already in Section 1.2, the iteration (6.1.13) may not converge even if the initial value x_0 is chosen arbitrarily close to a root. If $\lim_{n\to\infty} x_n = \alpha$ for all x_0 in a sufficiently close neighborhood of α the α is called a **point of attraction** otherwise α is a **point of repulsion**.

We shall see that under certain conditions the fixed-point problem has a unique solution and that the iteration defined by (6.1.13) converges to this solution. A sufficient condition for (6.1.13) to generate a convergent sequence is given in the following theorem.

Theorem 6.1.5.

Suppose that the function $\phi(x)$ has a real fixed point α, and that in the closed interval

$$J = \{x \mid |x-\alpha| \le \rho\}$$

$x \to \phi(x)$ is a **contraction mapping**, *i.e.,*

$$|\phi(s) - \phi(t)| \le C|s-t|, \quad 0 \le C < 1, \tag{6.1.16}$$

for arbitrary points s and t in J. Then for all $x_0 \in J$ the fixed-point iteration $x_n = \phi(x_{n-1})$, generates a sequence $\{x_n\}$ such that:

(a) *$x_n \in J$, $n = 1, 2, \ldots$;*

(b) *$\lim_{n\to\infty} x_n = \alpha$,*

(c) *α is the only root in J of $x = \phi(x)$.*

Proof. We first prove assertion (a), by induction. Suppose that $x_{n-1} \in J$. Then by (6.1.16) it follows that

$$|x_n - \alpha| = |\phi(x_{n-1}) - \phi(\alpha)| \le C|x_{n-1} - \alpha| \le C\rho.$$

Hence $x_n \in J$ and (a) is proved. Repeated use of the inequality above gives

$$|x_n - \alpha| \le C|x_{n-1} - \alpha| \le \cdots \le C^n|x_0 - \alpha|,$$

and since $C < 1$, the result (b) follows. Suppose, finally, that $x = \phi(x)$ has another root $\beta \in J$, $\beta \neq \alpha$. Then, by (6.1.16)

$$|\alpha - \beta| = |\phi(\alpha) - \phi(\beta)| < |\alpha - \beta|,$$

a contradiction; thus (c) follows. $\square$

Observe that the contractive Lipschitz condition (6.1.16) implies the continuity of ϕ. If $\phi'(x)$ exists, then a sufficient condition for (6.1.16) to hold is that

$$|\phi'(x)| \le C < 1, \quad \forall\, x \in J, \tag{6.1.17}$$

since then by the mean value theorem we have for $x, y \in J$ that

$$|\phi(x) - \phi(y)| = |\phi'(\zeta_n)||x - y| < |x - y|, \quad \zeta_n \in J.$$

On the other hand if $|\phi'(\alpha)| > 1$ then the iterative method (6.1.13) diverges. The four different cases that occur, depending on the sign and magnitude of $\phi'(\alpha)$ were illustrated in Figures 1.2.1a–d.

There is an analogue result valid for functions $\phi(z)$ of a complex variable z. Assume that $\phi(z)$ is defined and analytic in the circle

$$K = \{z \mid |z - \alpha| \leq \rho\},$$

where α is a fixed point. Then if $|\phi'(z)| \leq C < 1$, for all $z \in K$, the fixed point iteration

$$z_0 \in K, \qquad z_n = \phi(z_{n-1}), \quad n = 1, 2, \dots$$

converges to α, which is the only fixed point in K.

In Theorem 6.1.5 *we assumed the existence of a fixed point* α of $\phi(x)$. It is remarkable that the theorem can be modified so that it can be used to *prove the existence of a fixed point*, and hence of a root of the equation $x = \phi(x)$.

Theorem 6.1.6.

Let x_0 *be a starting point, and consider the fixed point iteration* $x_{n+1} = \phi(x_n)$, $n = 1, 2, \dots$. *Assume that* J *is a closed interval such that* $x_0 \in J$ *and*

$$|\phi(s) - \phi(t)| \leq C|s - t|, \quad 0 \leq C < 1, \tag{6.1.18}$$

for all $s, t \in J$. *Then if*

$$x_1 + \frac{C}{1 - C}(x_1 - x_0) \in J, \tag{6.1.19}$$

(a), (b) *and* (c) *of Theorem* 6.1.5 *are true.*

Proof. The theorem will be proved in a more general setting in Vol. II, Chapter 11 (see Theorem 11.2.1). □

We remark that (6.1.18) is satisfied if $|\phi(x)| \leq C < 1$ in J. Further, there is an analogue of this theorem also for complex functions $\phi(z)$ analytic in a circle $K = \{z \mid |z - \alpha| \leq \rho\}$ containing the initial approximation z_0.

An estimate of the error in x_n, which depends only on x_0, x_1 and the Lipschitz constant m, may be derived as follows. For arbitrary positive integers m and n we have

$$x_{m+n} - x_n = (x_{m+n} - x_{m+n-1}) + \cdots (x_{n+2} - x_{n+1}) + (x_{n+1} - x_n).$$

From the Lipschitz condition we conclude that $|x_{i+1} - x_i| \leq C^i|x_1 - x_0|$, and hence

$$|x_{m+n} - x_n| \leq (C^{m-1} + \cdots + C + 1)|x_{n+1} - x_n|.$$

Summing the geometric series and letting $m \to \infty$ we obtain

$$|\alpha - x_n| \leq \frac{1}{1-C}|x_{n+1} - x_n| \leq \frac{C^n}{1-C}|x_1 - x_0|. \tag{6.1.20}$$

Note that if C is close to unity, then the error in x_n can be much larger than $|x_{n+1} - x_n|$. Clearly it is not always safe to terminate the iterations when $|x_{n+1} - x_n|$ is less than the required tolerance!

Example 6.1.7.

For a linearly convergent fixed point iteration the sequence $\{x_j - \alpha\}$ approximately forms a geometric series. Then, as seen in Sec. 3.3.2, a more rapidly convergent sequence $\{x'_j\}$ can be obtained by Aitken extrapolation,

$$x'_j = x_j - (\nabla x_j)^2/\nabla^2 x_j. \tag{6.1.21}$$

Note that if the convergence is *not* linear, then the sequence $\{x'_n\}$ will usually converge *slower* than $\{x_n\}$!

The equation $x = e^{-x}$ has one root $\alpha \approx 0.567$. Using the fixed point iteration $x_{n+1} = e^{-x_n}$ combined with Aitken extrapolation we obtain the result shown in the table below.

j	x_j	∇x_j	$\nabla^2 x_j$	x'_j
0	**0.567**00 00000			
1	**0.567**22 45624	2245624		
2	**0.567**09 71994	-1273630	-3519254	**0.56714 329**25
3	**0.5671**6 94312	722318	1995948	**0.56714 329**11

It is seen that in this example the extrapolated sequence $\{x'_j\}$ converges much more rapidly, and nine correct decimals are obtained.

In the above example Aitken extrapolation was used in a *passive* way to transform the sequence $\{x_n\}$ into $\{x'_j\}$. It is also possible to use Aitken extrapolation in an *active way* (cf. Example 3.3.6). We start as before by computing $x_1 = \phi(x_0)$, $x_2 = \phi(x_1)$ and apply the formula (6.3.22) to compute x'_2. Next we continue the iterations from x'_2, i.e., compute $x_3 = \phi(x'_2)$, $x_4 = \phi(x_3)$. We can now extrapolate from x'_2, x_3 and x_4 to get x'_4, etc. It is easily verified that the sequence $z_n = x'_{2n}$ is generated by the fixed-point iteration

$$z_{n+1} = \psi(z_n), \quad \psi(z) = z - \frac{(\phi(z) - z)^2}{(\phi(\phi(z)) - \phi(z)) - (\phi(z) - z)}.$$

This iteration may converge even when the basic iteration $x_{n+1} = \phi(x_n)$ diverges!

6.1.5 Convergence Order and Efficiency

In general we will be given an equation $f(x) = 0$ to solve and want to construct a fixed point iteration such converges rapidly. Basic concepts to quantify the rate of convergence will now be introduced.

Definition 6.1.7.

Consider a sequence $\{x_n\}_0^\infty$ *with* $\lim_{n\to\infty} x_n = \alpha$, *and* $x_n \neq \alpha$ *for* $n < \infty$. *The sequence is said to have* **convergence order** *equal to* $q \geq 1$ *if for some constant* $0 < C < \infty$ *it holds that*

$$\lim_{n\to\infty} \frac{|x_{n+1} - \alpha|}{|x_n - \alpha|^q} = C. \tag{6.1.22}$$

Here C *is called the* **asymptotic error constant**.

If $q = 1$ *then we require that* $C < 1$ *and then* $\{x_n\}$ *is said to converge* **linearly** *and* C *is the rate of linear convergence. For* $q = 2, 3$ *the convergence is called* quadratic, *and* cubic, *but* q *need not be an integer.*

More precisely, the order q in Theorem 6.1.7 is called the Q-order of convergence, where Q stands for quotient. The same definitions can be used also for vector-valued sequences. Then absolute values in (6.1.22) are replaced by a vector norm.

There are types of convergence that are not covered by the above definition of order. A sequence may converge more slowly than linear so that (6.1.22) holds with $q = 1$ and $C = 1$. Then convergence is called **sublinear**. If (6.1.22) holds with $q = 1$ and $C = 0$, but not for any value of $q > 1$ then convergence is called **superlinear**.

Example 6.1.8.

Examples of sublinear, linear and superlinear convergence are

$$x_n = 1/n, \quad x_n = 2^{-n}, \quad \text{and} \quad x_n = n^{-n},$$

respectively.

Alternative definitions of convergence order are considered by Ortega and Rheinboldt [18, Chap. 9] and Brent [2, Sec. 3.2]. For example, if

$$\lim_{n\to\infty} \inf(-\log|x_n - \alpha|)^{1/n} = q, \tag{6.1.23}$$

then q is called weak order of convergence for x_n, since (6.1.22) implies (6.1.23), but not vice versa. For example, the sequence $x_n = \exp(-p^n)(2 + (-1)^n)$ converges to 0 with weak order p. However, the limit in (6.1.22) does not exist if $q = p$, is zero if $q < p$ and infinite if $q > p$.

Consider a fixed point iteration $x_{n+1} = \phi(x_n)$. Assume that $\phi'(x)$ exists and is continuous in a neighborhood α. It then follows from the proof of Theorem 6.1.5, that if $0 < |\phi'(\alpha)| < 1$ and x_0 is chosen sufficiently close to α, then the sequence x_n generated by $x_{n+1} = \phi(x_n)$ satisfies (6.1.22) with $q = 1$ and $C = |\phi'(\alpha)|$.

The number of accurate decimal places in the approximation x_n equals $\delta_n = -\log_{10}|x_n - \alpha|$. Equation (6.1.22) implies that

$$\delta_{n+1} \approx q\delta_n - \log_{10}|C|.$$

Hence for linear convergence ($q = 1$) as $n \to \infty$ each iteration gives a fixed (fractional) number of additional decimal places. For a method with convergence of order $q > 1$ each iteration increases the number of correct decimal places q-fold as $n \to \infty$. This shows that eventually a method with larger order of convergence will converge faster.

Example 6.1.9.

Consider a sequence x_n with quadratic convergence with $C = 1$. Set $\epsilon_n = |x_n - \alpha|$ and assume that $\epsilon_0 = 0.9$. From $\epsilon_{n+1} \le C\epsilon_n^2$, it follows that ϵ_n, for $n = 2, 3, \dots$, is bounded by

$$0.81, 0.66, 0.43, 0.19, 0.034, 0.0012, 1.4 \cdot 10^{-6}, 1.9 \cdot 10^{-12}, \dots,$$

respectively. For $n \ge 6$ the number of significant digits is approximately doubled at each iteration!

Consider an iteration method with convergence order $q \ge 1$. If each iteration requires m units of work (usually the work involved in computing a function value or a value of one of its derivatives) then the **efficiency index** of the iteration is defined as

$$E = q^{1/m}. \tag{6.1.24}$$

The efficiency index gives a basis for comparing the efficiency of iterative methods of different order of superlinear convergence. Assuming that the cost of evaluating $f(x_n)$ and $f'(x_n)$ is two units the efficiency index for Newton's method is $E = 2^{1/2} = \sqrt{2}$. (Methods that converge linearly all have $E = 1$.)

The order of the fixed-point iteration $x_{n+1} = \phi(x_n)$ can be determined if $\phi(x)$ is sufficiently many times continuously differentiable in a neighborhood of α.

Theorem 6.1.8. *Assume that $\phi(x)$ is p times continuously differentiable. Then the iteration method $x_{n+1} = \phi(x_n)$ is of order p for the root α if and only if*

$$\phi^{(j)}(\alpha) = 0, \quad j = 1 : p - 1, \quad \phi^{(p)}(\alpha) \ne 0. \tag{6.1.25}$$

Proof. If equation (6.1.25) holds, then according to Taylor's theorem we have

$$x_{n+1} = \phi(x_n) = \alpha + \frac{1}{p!}\phi^{(p)}(\zeta_n)(x_n - \alpha)^p, \quad \zeta_n \in \operatorname{int}(x_n, \alpha).$$

Hence for a convergent sequence x_n the error $\epsilon_n = x_n - \alpha$ satisfies

$$\lim_{n\to\infty} |\epsilon_{n+1}|/|\epsilon_n|^p = |\phi^{(p)}(\alpha)|/p! \ne 0,$$

and the order of convergence equals p. It also follows that if $\phi^{(j)}(\alpha) \ne 0$ for some j, $1 \le j < p$, or if $\phi^{(p)}(\alpha) = 0$, then the iteration cannot be of order p. $\square$

Example 6.1.10.

We remarked before that to compute a root α of $f(x) = 0$, we can use a fixed point iteration with $\phi(x) = x - f(x)g(x)$, where $g(x)$ is an arbitrary function such that $g(\alpha) \neq 0$. we evaluate the derivative

$$\phi'(x) = 1 - f'(x)g(x) - f(x)g'(x).$$

To achieve quadratic convergence we take $g(x) = 1/f'(x)$. Assuming that $f'(\alpha =\neq 0$ we find, using $f(\alpha) = 0$, that $\phi'(\alpha) = 1 - f'(\alpha)g(\alpha) = 0$, Hence the iteration

$$x_{n+1} = x_n - f(x_n)/f'(x_n), \tag{6.1.26}$$

achieves quadratic convergence. This is Newton's method, which will be treated at length in Sec. 6.3

Review Questions

1. What does limit the final accuracy of a root computed by the bisection algorithm? Discuss suitable termination criteria.

2. (a) Given a nonlinear scalar equation $f(x) = 0$ with a simple root α. How can a fixed point iteration $x_{n+1} = \phi(x_n)$ be constructed, which converges to α?

(b) Assuming that a fixed point α exists for the mapping $x = \phi(x)$. Give sufficient conditions for convergence of the sequence generated by $x_{n+1} = \phi(x_n)$.

(c) How can the conditions in (b) be modified so that the *existence* of a fixed point can be proved?

3. (a) Define the concepts order of convergence and asymptotic error constant for a convergent sequence $\{x_n\}$ with $\lim_{n\to\infty} x_n = \alpha$.

(b) What is meant by sublinear and superlinear convergence? Give examples of sequences with sublinear and superlinear convergence.

4. (a) Define the efficiency index of a given iterative method of order p and asymptotic error constant $C \neq 0$.

(b) Determine the order of a new iterative method consisting of m consecutive steps of the method in (a). What is the order and error constant of this new method? Show that it has the same efficiency index as the first method.

5. (a) When can (passive) Aitken extrapolation be applied to speed up the convergence of sequence.

(b) Describe the difference between active and passive Aitken extrapolation?

6. What two quantities determines the attainable accuracy of a simple root α to the equation $f(x) = 0$. Give an example of an ill-conditioned root.

7. Discuss the choice of termination criteria for iterative methods.

Problems and Computer Exercises

1. Use graphic representation to determine the zeros of the following functions to one correct decimal:

(a) $4\sin x + 1 - x$; (b) $1 - x - e^{-2x}$; (c) $(x+1)e^{x-1} - 1$;

(d) $x^4 - 4x^3 + 2x^2 - 8$; (e) $e^x + x^2 + x$; (f) $e^x - x^2 - 2x - 2$;

(g) $3x^2 + \tan x$.

2. Show analytically that the equation $xe^{-x} = \gamma$ has exactly two real roots when $\gamma < e^{-1}$.

3. Plot the functions $f(x) = \cosh x$ and $g(x) = 1/\cos x$ and deduce that the equation $\cosh x \cos x = 1$ has its smallest positive root in the interval $(3\pi/2, 2\pi)$. Determine this root using the bisection method.

4. The following equations all have a root in the interval $(0, 1.6)$ Determine these with an error less than 10^{-8} using the bisection method.

(a) $x\cos x = \ln x$; (b) $2x = e^{-x}$; (c) $e^{-2x} = 1 - x$.

5. Locate the real root of the equation

$$e^x(x-1) = e^{-x}(x+1),$$

by graphing both sides. Then compute the root with an error less than 10^{-8} using bisection. How many bisection steps are needed?

6. Let k be a given non-negative number and consider the equation $\sin x = -k\cos x$. This equation has infinitely many roots. Separate the roots, i.e., partition the real axis into intervals which contain exactly one root.

7. The choice of m_k as the *arithmetic* mean of a_{k-1} and b_{k-1} in the bisection method minimizes the worst case maximum *absolute* error. If in the case that $ab > 0$ we take instead

$$m_k = \sqrt{a_k b_k}$$

i.e., the *geometric* mean, then the worst case *relative* error is minimized. Do Example 6.1.2 using this variation of the bisection method.

8. In Example 6.1.6 three different fixed point iterations were suggested for solving the equation $x + \ln x = 0$. (a) Which of the formulas *can* be used?

(b) Which of the formulas *should* be used?

(c) Give an even better formula!

9. Investigate if and to what limit the iteration $x_{n+1} = 2^{x_n - 1}$ sequence converges for various choices of x_0.

10. (L. Wittmeyer-Koch) (a) A fixed point iteration $x_{n+1} = \phi(x_n)$ can converge also when $|\phi'(\alpha)| = 1$. Verify this by graphing the iteration for $\phi(x) = x + (x-1)^2$, $x_0 = 0.6$. which has the fixed point $\alpha = 1$.

(b) Show that for the iteration in (a) if $x_n = 1 - \epsilon$, then

$$\frac{|x_{n+1} - 1|}{|x_n - 1|} = 1 - \epsilon,$$

i.e. the asymptotic rate of convergence is sublinear.

11. (a) Consider the fixed point iteration $x_{n+1} = \phi(x_n)$, where $\phi(x) = x+(x-1)^2$. Show that this has a fixed point for $\alpha = 1$ and that $\phi'(\alpha) = 1$.

(b) Show that the iteration in (a) is convergent for $x_0 < 1$.

12. Consider the iteration $x_{n+1} = 1-\lambda x_n^2$. Illustrate graphically how the iteration for $\lambda = 0.7, 0.9, 2$. (For $\lambda = 2$ the iteration is chaotic.)

13. Use active Aitken extrapolation on the (divergent) iterative method $x_{n+1} = 5\ln x_n$ to compute the smallest root of the equation $x = 5\ln x$. Start with $x_0 = 1.3$.

6.2 Methods Based on Interpolation

6.2.1 The Method of False Position

Assume that we have two initial approximations $a_0 = a$ and $b_0 = b$ such that $f(a)f(b) < 0$. As in the bisection method we generate a nested sequence of intervals $(a_0, b_0) \supset (a_1, b_1) \supset (a_2, b_2) \subset \cdots$ such that $f(a_n)f(b_n) < 0$, $n = 0, 1, 2, \ldots$. Given (a_n, b_n), we take x_{n+1} to be the intersection of the secant through the point $(a_n, f(a_n))$ and $(b_n, f(b_n))$. Then by Newton's interpolation formula x_{n+1} satisfies

$$0 = f(a_n) + (x_{n+1} - a_n)\frac{f(a_n) - f(b_n)}{a_n - b_n}$$

giving

$$x_{n+1} = a_n - f(a_n)\frac{a_n - b_n}{f(a_n) - f(b_n)}. \qquad (6.2.1)$$

If $f(x_{n+1})f(a_n) > 0$, set $a_{n+1} = x_{n+1}$ and $b_{n+1} = b_n$; otherwise set $b_{n+1} = x_{n+1}$ and $a_{n+1} = a_n$. This is the **false-position method** or in Latin **regula falsi**.[2] Note that if $f(x)$ is linear we obtain the root in just one step, but sometimes the rate of convergence can be much slower than for bisection.

Suppose now that $f(x)$ is convex on $[a, b]$, $f(a) < 0$, and $f(b) > 0$, as in Figure 6.2.1. Then the secant through $x = a$ and $x = b$ will lie above the curve and hence intersect the x-axis to the left of α. The same is true for all subsequent secants and therefore the right endpoint b will be kept. The approximations $x_1, x_2, x_3, \ldots$ will all lie on the convex side of the curve and cannot go beyond the root α. A similar behavior, with monotone convergence and one of the points a or b fixed, will occur whenever $f''(x)$ exists and has constant sign on $[a, b]$.

Example 6.2.1.

We apply the method of false position to the $f(x) = (x/2)^2 - \sin x = 0$ from Example 6.1.2 with initial approximations $a_0 = 1.5$, $b_1 = 2$. We have $f(1.5) = -0.434995 < 0$ and $f(2.0) = +0.090703 > 0$ and successive iterates are

[2]The method of regula falsi is very old, originating in 5th century Indian texts, and was used in medieval Arabic mathematics. It got its name from the Italian mathematician Leonardi Pisano in the 13th century.

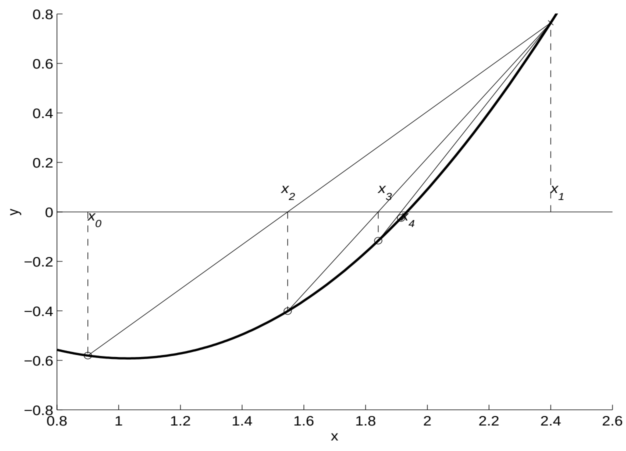

Figure 6.2.1. *The false-position method.*

n	x_n	$f(x_n)$	h_n
1	**1.9**13731 221035	−0.026180060742	−0.019322989205
2	**1.933**054 210240	−0.000924399645	−0.000675397892
3	**1.9337**29 608132	−0.000031930094	−0.000023321005
4	**1.93375**2 929137	−0.000001102069	−0.000000804916
5	**1.933753 7**34053		

Note that $f(x_n) < 0$ for all $n \geq 0$ and consequently $b_n = 2$ is fixed. In the limit convergence is linear with rate approximately equal to $C \approx 0.034$.

If f is twice continuously differentiable and $f''(\alpha) \neq 0$, then eventually an interval will be reached on which $f''(x)$ does not change sign. Then, as in the example above, one of the endpoints (say b) will be retained and $a_n = x_n$ in all future steps. By (6.2.1) the successive iterations are

$$x_{n+1} = x_n - f(x_n)\frac{x_n - b}{f(x_n) - f(b)}.$$

To determine the speed of convergence subtract α and divide by $\epsilon_n = x_n - \alpha$ to get

$$\frac{\epsilon_{n+1}}{\epsilon_n} = 1 - \frac{f(x_n)}{x_n - \alpha}\frac{x_n - b}{f(x_n) - f(b)}.$$

Since $\lim_{n\to\infty} x_n = \alpha$ and $f(\alpha) = 0$, it follows that

$$\lim_{n\to\infty} \frac{\epsilon_{n+1}}{\epsilon_n} = C = 1 - (b - \alpha)\frac{f'(\alpha)}{f(b)}, \tag{6.2.2}$$

which shows that convergence is linear. Convergence will be very slow if $f(x)$ is very flat near the root α, $f(b)$ is large, and α near b since then $(b-\alpha)f'(\alpha) \ll f(b)$ and $C \approx 1$.

6.2.2 The Secant Method

A serious drawback with the method of false position is that ultimately one endpoint of the sequence of bracketing intervals will become fixed and therefore the length $(b_n - a_n)$ will not tend to zero. This can be avoided and convergence substantially improved by always using the secant through *the last two points* $(x_{n-1}, f(x_{n-1}))$ and $(x_n, f(x_n))$. The next approximation x_{n+1} is determined as the abscissa of the point of intersection between this secant and the x-axis; see Figure 6.2.2.

Given initial approximations $x_{-1} = a$ and $x_0 = b$, approximations $x_1, x_2, , x_3, \dots$ are computed by

$$x_{n+1} = x_n + h_n, \quad h_n = -f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}, \quad n \geq 1, \tag{6.2.3}$$

assuming that $f(x_n) \neq f(x_{n-1})$. This is the **secant method**, which historically predates Newton's method.

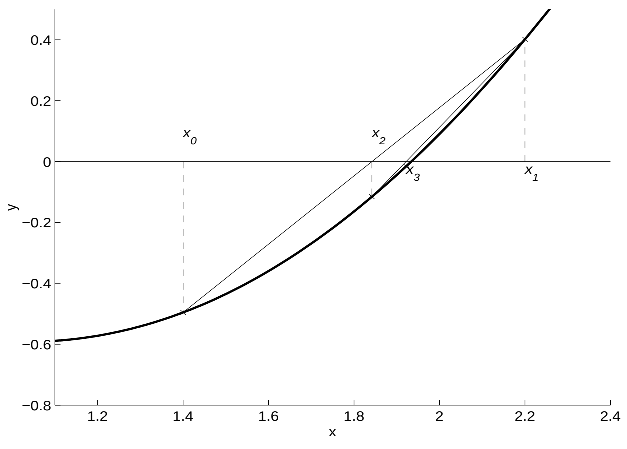

Figure 6.2.2. *The secant method.*

Notice that although regula falsi and the secant method require two initial approximations to the root, only *one function evaluation per step* is needed. The iteration, which is of the form $x_{n+1} = \phi(x_n; x_{n-1})$, is not a fixed point iteration as defined in Section 6.1.4. Sometimes methods of this form, which use old information at x_{n_1}, are called fixed point iterations with memory.

When the secant method converges $|x_n - x_{n-1}|$ will eventually become small. The quotient $(x_n - x_{n-1})/(f(x_n) - f(x_{n-1}))$ will then be determined with poor relative accuracy. If x_n and x_{n-1} both are very close to the root α and not bracketing α, then the resulting rounding error in x_{n+1} can then become very large. Fortunately, from the error analysis below it follows that the approximations in general are such that $|x_n - x_{n-1}| \gg |x_n - \alpha|$ and the dominant contribution to the round-off error in x_{n+1} comes from the error in $f(x_n)$. Note that (6.2.3) should *not* be rewritten in the form

$$x_{n+1} = \frac{x_{n-1}f(x_n) - x_n f(x_{n-1})}{f(x_n) - f(x_{n-1})},$$

since this formula can give rise to severe difficulties with *cancellation* when $x_n \approx x_{n-1}$ and $f(x_n)f(x_{n-1}) > 0$. Even (6.2.3) is not always safe to use. We must take care to avoid overflow or division by zero. Without restriction we can assume that $|f(x_{n-1})| \geq |f(x_n)| > 0$ (otherwise renumber the two points). Then, s_n can be computed without risk of overflow from

$$x_{n+1} = x_n + \frac{s_n}{1 - s_n}(x_n - x_{n-1}), \quad s_n = \frac{f(x_n)}{f(x_{n-1})}. \tag{6.2.4}$$

where the division with $1 - s_n$ is only carried out if $1 - s_n$ is large enough.

Example 6.2.2.

To illustrate the improved convergence of the secant method we consider once again the equation $f(x) = (x/2)^2 - \sin x = 0$ with initial approximations $x_0 = 1.5$, $x_1 = 2$. The result is:

n	x_n	$f(x_n)$	h_n
-1	1.5	−0.434994 986604	
0	2.0	+0.090702 573174	−0.086268 778965
1	**1.9**13731 221035	−0.026180 060742	+0.019322 989205
2	**1.933**054 210240	−0.000924 399645	+0.000707 253882
3	**1.9337**61 464122	+0.000010 180519	−0.000007 704220
4	**1.933753 75**9902	−0.000000 003867	+0.000000 002925
5	**1.933753 762827**		

Note that the approximations x_1 and x_2 are the same as for the false position. method, but here x_4 is correct to eight decimals and x_5 to twelve decimals. The rapid convergence is partly because $x_1 = 2$ is quite a good initial approximation. However, note that although the root is bracketed by the initial intervals $[x_0, x_1]$ and $[x_1, x_2]$ it lies outside the interval $[x_2, x_3]$.

Assume that f is twice continuously differentiable. Then according to Newton's interpolation formula with error term (Theorem 4.3.1) we have

$$f(x) = f(x_n) + (x - x_n)[x_{n-1}, x_n]f + (x - x_{n-1})(x - x_n)\frac{f''(\zeta_n)}{2}, \tag{6.2.5}$$

where $\zeta_n \in \operatorname{int}(x, x_{n-1}, x_n)$

$$f[x_{n-1}, x_n] = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}.$$

To derive an asymptotic formula for the secant method we put $x = \alpha$ in (6.2.5) and subtract the secant equation $0 = f(x_n) + (x_{n+1} - x_n)[x_{n-1}, x_n]f$. Since $f(\alpha) = 0$ we get

$$(\alpha - x_{n+1})[x_{n-1}, x_n]f + (\alpha - x_{n-1})(\alpha - x_n)f''(\zeta_n)/2 = 0,$$

where $\zeta_n \in \text{int}(\alpha, x_{n-1}, x_n)$. According to the mean-value theorem, we have

$$[x_{n-1}, x_n]f = f'(\zeta_n'), \quad \zeta_n' \in \text{int}(x_{n-1}, x_n),$$

and it follows that

$$\epsilon_{n+1} = \frac{1}{2}\frac{f''(\zeta_n)}{f'(\zeta_n')}\epsilon_n\epsilon_{n-1}. \tag{6.2.6}$$

Example 6.2.3.

The ratios $\epsilon_{n+1}/(\epsilon_n\epsilon_{n-1})$ in Example 6.2.2 are equal to 0.697, 0.527, 0.550, $n = 1:3$, which compares well with the limiting value 0.543 of $\frac{1}{2}f''(\alpha)/f'(\alpha)$.

From (6.2.6) it can be deduced that the secant method always converges from starting values x_0, x_1 sufficiently close to α. For this to be true it suffices that the first derivative $f'(x)$ is continuous, since then

$$\epsilon_{n+1} = \left(1 - \frac{f'(\xi_n)}{f'(\zeta_n)}\right)\epsilon_n, \quad \xi_n \in \text{int}(x_{n-1}, \alpha), \quad \zeta_n \in \text{int}(x_n, x_{n-1}).$$

However, *in the secant method there is no guarantee that the computed sequence of approximations stay in the initial interval* $[x_0, x_1]$. Unlike the steady convergence of the bisection method things can go seriously wrong using the secant method! A remedy will be discussed in Sec. 6.2.4.

The following theorem gives the order of convergence for the secant method.

Theorem 6.2.1. *Suppose that $f(x)$ is twice continuously differentiable and that in a neighborhood I of the root α, containing $x_0, x_1, x_2, \dots, x_n$, we have*

$$\frac{1}{2}\left|\frac{f''(y)}{f'(x)}\right| \le M, \quad x, y \in I.$$

Let $q = (1+\sqrt{5})/2 = 1.618\dots$ be the unique positive root of the equation $\mu^2 - \mu - 1 = 0$ and set

$$K = \max\big(M|\epsilon_0|, (M|\epsilon_1|)^{1/q}\big), \quad n = 0, 1, 2, \dots. \tag{6.2.7}$$

Then it holds that

$$|\epsilon_n| \le \frac{1}{M}K^{q^n}, \tag{6.2.8}$$

i.e., the iteration has convergence order q.

Proof. The proof is by induction. From the choice of K it follows that (6.2.8) is trivially true for $n = 0, 1$. Suppose that (6.2.8) holds for n and $n-1$. Then since $q^2 = q + 1$ it follows using the assumption and (6.2.6) that

$$|\epsilon_{n+1}| \le M|\epsilon_n||\epsilon_{n-1}| \le \frac{1}{M}K^{q^n}K^{q^{n-1}} = \frac{1}{M}K^{q^n + q^{n-1}} = \frac{1}{M}K^{q^{n+1}}. \tag{6.2.9}$$

□

To compare the efficiency of the secant method and Newton's method, which is quadratically convergent, we use the efficiency index introduced in Section 6.1.5. Assume that the work to compute $f'(x)$ is θ times the amount of work required to compute $f(x)$. Then, with the same amount of work we can perform $k(1+\theta)$ iterations with the secant method and k iterations with Newton's method. Equating the errors we get $(m\epsilon_0)^{2^k} = (m\epsilon_0)^{p^{k(1+\theta)}}$, where $q = 1.618\ldots$. Hence the errors are the same for both methods when $p^{k(1+\theta)} = 2^k$ or

$$(1+\theta)\log\left(\tfrac{1}{2}(1+\sqrt{5})\right) = \log 2,$$

which gives $\theta = 0.4404\ldots$. Thus, from this analysis we conclude that if $\theta > 0.44$, then the secant method is asymptotically more efficient than Newton's method.

In Example 6.2.2 we can observe that the error $\epsilon_n = x_n - \alpha_n$ changes sign at every third step. Hence, in this example,

$$\alpha \in \operatorname{int}(x_{n+1} - x_n), \quad n = 0, 1, 3, 4, \ldots.$$

That is, *the root α is bracketed by x_n and x_{n+1} except for every third step.* We shall show that this is no coincidence. Assume that $x_n \in (a, b)$, $n = 0, 1, 2, \ldots$, and that $f'(x) \neq 0$ and $f''(x)$ does not change sign in (a, b). Then from (6.2.6) it follows that the ration

$$\frac{\epsilon_{n+1}}{\epsilon_n \epsilon_{n-1}}$$

will have constant sign for all n. Then if $\alpha \in \operatorname{int}(x_0, x_1)$ and $\epsilon_0\epsilon_1 < 0$, and it follows that the sign of ϵ_n must change every third step according to one of the following two schemes (verify this!):

$$\cdots + - + + - + + - + + \cdots$$
$$\cdots + - - + - - + - - + \cdots$$

Hence convergence for the secant method, if it occurs, will take place in a waltz rhythm! This means that at every third step the last two iterates x_{n-1} and x_n will not always bracket the root.

6.2.3 Higher Order Interpolating Methods

In the secant method linear interpolation through (x_{n-1}, f_{n-1}) and (x_n, f_n) is used to determine the next approximation to the root. A natural generalization is to use an interpolating method of higher order. Let $x_{n-r}, \ldots, x_{n-1}, x_n$ be $r+1$ distinct approximations and determine the (unique) polynomial $p(x)$ of degree r interpolating $(x_{n-j}, f(x_{n-j}))$, $j = 0 : r$. By Newton's interpolation formula (Sec. 4.2.1) the interpolating polynomial is

$$p(x) = f_n + [x_n, x_{n-1}]f \cdot (x - x_n) + \sum_{j=2}^{r} [x_n, x_{n-1}, \ldots, x_{n-j}]f \; \Phi_j(x),$$

where

$$\Phi_j(x) = (x - x_n)(x - x_{n-1}) \cdots (x - x_{n-j}).$$

The next approximation x_{n+1} is taken as the real root to the equation $p(x) = 0$ closest to x_n and x_{n-r} is deleted. Suppose the interpolation points lie in an interval J, which contains the root α and in which $f'(x) \neq 0$. It can be shown that if there is at least one interpolation point on each side of α then $p(x) = 0$ has a real root in J. Further the following formula for the error holds (Traub [25, pp. 67–75])

$$\epsilon_{n+1} = -\frac{f^{(r+1)}(\zeta_n)}{(r+1)!p'(\eta_n)} \prod_{i=0}^{r} \epsilon_{n-i}, \tag{6.2.10}$$

where $\zeta_n \in \operatorname{int}(\alpha, x_{n-1}, x_n)$ and $\eta_n \in \operatorname{int}(\alpha, x_{n+1})$.

In the special case $r = 2$ we get the quadratic equation

$$p(x) = f_n + (x - x_n)[x_n, x_{n-1}]f + (x - x_n)(x - x_{n-1})[x_n, x_{n-1}, x_{n-2}]f. \tag{6.2.11}$$

We assume that $[x_n, x_{n-1}, x_{n-2}]f \neq 0$ since otherwise the method degenerates into the secant method. Setting $h_n = (x - x_n)$ and writing $(x - x_{n-1}) = h_n + (x_n - x_{n-1})$, this equation becomes

$$h_n^2[x_n, x_{n-1}, x_{n-2}]f + \omega h_n + f_n = 0, \tag{6.2.12}$$

where

$$\omega = [x_n, x_{n-1}]f + (x_n - x_{n-1})[x_n, x_{n-1}, x_{n-2}]f. \tag{6.2.13}$$

The root closest to x_n corresponds to the root h_n of smallest absolute value to the equation (6.2.12). To express this root in a numerically stable way the standard formula for the roots of a quadratic equation should be multiplied by its conjugate quantity (see Example 2.3.3). Using this formula we get

$$x_{n+1} = x_n + h_n, \quad h_n = -\frac{2f_n}{\omega \pm \sqrt{\omega^2 - 4f_n\,[x_n, x_{n-1}, x_{n-2}]f}}, \tag{6.2.14}$$

where the sign in the denominator should be chosen so as to minimize $|h_n|$. This is the **Muller–Traub method**.

A drawback is that the equation (6.2.12) may not have a real root even if a real zero is being sought. On the other hand, this means that the Muller–Traub method has the useful property that complex roots may be found from real starting approximations.

By (6.2.10) it follows that

$$\epsilon_{n+1} = -\frac{f'''(\zeta_n)}{3!\,p'(\eta_n)} \epsilon_n \epsilon_{n-1} \epsilon_n. \tag{6.2.15}$$

It can be shown that the convergence order for the Muller–Traub method is at least $q = 1.839\ldots$, which equals the largest root of the equation $\mu^3 - \mu^2 - \mu - 1 = 0$ (cf.Theorem 6.2.1). Hence this method does not quite achieve quadratic convergence. In fact, it can be shown under very weak restrictions that *no iterative method using only one function evaluation can have* $q \geq 2$.

For $r > 2$ there are no useful explicit formulas for determining the zeros of the interpolating polynomial $p(x)$. Then we can proceed as follows. We write the equation $p(x) = 0$ in the form $x = x_n + F(x)$, where

$$F(x) \equiv \frac{-f_n - \sum_{j=2}^{r} [x_n, x_{n-1}, \dots x_{n-j}]f \, \Phi_j(x)}{[x_n, x_{n-1}]f}$$

(cf. Sec. 4.2.4). Then a fixed point *iteration* can be used to solve for x. To get the first guess x^0 we ignore the sum (this means using the secant method) and then iterate, $x^i = x_n + F(x^{i-1})$, $i = 1, 2, \dots$ until x^i and x^{i-1} are close enough.

Suppose that $x_{n-j} - x_n = O(h)$, $j = 1 : r$, where h is some small parameter in the context (usually some step size). Then $\Phi_j(x) = O(h^j)$, $\Phi_j'(x) = O(h^{j-1})$. The divided differences are $O(1)$, and we assume that $[x_n, x_{n-1}]f$ is bounded away from zero. Then the terms of the sum decrease like h^j. The convergence ratio $F'(x)$ is here approximately

$$\frac{\Phi_2'(x)[x_n, x_{n-1}, x_{n-2}]f}{[x_n, x_{n-1}]f} = O(h).$$

So, if h is small enough, the iterations converge rapidly.

A different way to extend the secant method is to use **inverse interpolation**. Assume that $y_n, y_{n-1}, \dots, y_{n-r}$ are distinct and let $q(y)$ be the unique polynomial in y interpolating the values $x_n, x_{n-1}, \dots, x_{n-r}$. Reversing the rule of x and y and using Newton's interpolation formula this interpolating polynomial is

$$q(y) = x_n + [y_n, y_{n-1}]g \cdot (y - y_n) + \sum_{j=2}^{r} [y_n, y_{n-1}, \dots, y_{n-j}]f \, \Psi_j(y),$$

where $g(y_{n-j}) = x_{n-j}$, $j = 0 : r$.

$$\Psi_j(y) = (y - y_n)(y - y_{n-1}) \cdots (y - y_{n-j}).$$

The next approximation is then taken to be $x_{n+1} = q(0)$, that is

$$x_{n+1} = x_n - y_n \, [y_n, y_{n-1}]g + \sum_{j=2}^{r} [y_n, y_{n-1}, \dots, y_{n-j}]g \, \Psi_j(0),$$

For $r = 1$ there is no difference between direct and inverse interpolation and we recover the secant method. For $r > 1$ inverse interpolation as a rule gives different results. Inverse interpolation has the advantage of not requiring the solution of a polynomial equation. (For other ways of avoiding this see Problems 3 and 4.) The case $r = 2$ corresponds to inverse quadratic interpolation

$$x_{n+1} = x_n - y_n \, [y_n, y_{n-1}]g + y_n y_{n-1} \, [y_n, y_{n-1}, y_{n-2}]g, \tag{6.2.16}$$

This method has the same order of convergence as the Muller–Traub method.

Note that this method requires that y_n, y_{n-1}, and y_{n-2} are distinct. Even if this is the case it is not always safe to compute x_{n+1} from (6.2.16). Care has to be taken in order to avoid overflow and possibly division by zero. If we assume that $0 \neq |y_n| \leq |y_{n-1}| \leq |y_{n-2}|$ then it is safe to compute

$$s_n = y_n/y_{n-1}, \quad s_{n-1} = y_{n-1}/y_{n-2}, \quad r_n = y_n/y_{n-2} = s_n s_{n-1}.$$

We can rewrite (6.2.16) in the form $x_{n+1} = x_n + p_n/q_n$, where

$$\begin{aligned} p_n &= s_n[(1-r_n)(x_n - x_{n-1}) - s_{n-1}(s_{n-1} - r_n)(x_n - x_{n-2})], \\ q_n &= (1-s_n)(1-s_{n-1})(1-r_n). \end{aligned}$$

The final division p_n/q_n is only carried out if the correction is sufficiently small.

6.2.4 A Robust Hybrid Method

Efficient and robust root finders can be constructed by combining the secant method (or some higher order interpolation method) with bisection, A simple combination of Newton method with bisection will be discussed in Section 6.3.2.

A particularly elegant combination of bisection and the secant method was developed in the 1960s by van Wijngaarden, Dekker and others at the Mathematical Center in Amsterdam. A related algorithm, called zeroin, which combines bisection, the secant method and inverse quadratic interpolation, was developed by Brent [2]. The Matlab function "fzero", which finds a zero near a given approximation x_0, is based on Zeroin. A discussion of a slightly simplified version of fzero is given in Moler [16, Ch. 4.7].

We now outline the basic ideas used in zeroin. Start with a and b such that $f(a)f(b) < 0$ and use a secant step to get c in (a, b). We the repeat the following steps until $|b - a| < tol$ or $f(b) = 0$:

- Arrange a, b, and c so that $f(a)$ and $f(b)$ have opposite sign, $|f(b)| \leq |f(a)|$, and c is the value of b in the previous step.
- If $c \neq a$ compute the step using inverse quadratic interpolation; otherwise compute a secant step.
- If the computed step gives an approximation in $[a, b]$ take it; otherwise take a bisection step.

Review Questions

1. Sketch a function $f(x)$ with a root in (a, b), such that regula falsi converges very slowly.
2. Outline how the secant method can be safeguarded by combining it with the bisection method.
3. What property should the function $f(x)$ have to be unimodal on the interval $[a, b]$?

4. Discuss root finding methods based on quadratic interpolation (Muller–Traub's method) and inverse quadratic interpolation. What are the merits of these two approaches?

Problems and Computer Exercises

1. Use the secant method to determine the roots of the following equations to six correct decimals

   (a) $2x = e^{-x}$; (b) $\tan x + \cosh x = 0$.

2. Assume that we have $f_n f_{n-1} < 0$, and have computed x_{n+1}. If $f_{n+1}f_n < 0$ then in the next step we compute x_{n+2} by a secant step otherwise we use a line through (x_{n+1}, f_{n+1}) and $(x_{n-1}, \theta f_{n-1})$, where $0 < \theta < 1$. Clearly, $\theta = 1$ corresponds to a step with the method of false position and will usually give $f_{n+2}f_{n+1} > 0$. On the other hand, $\theta = 0$ gives $x_{n+1} = x_n$, and thus $f_{n+1}f_n < 0$. Hence a suitable choice of θ will always give $f_{n+2}f_{n+1} < 0$.

   Show that with $\theta = 0.5$ in a modified step it holds asymptotically $\epsilon_{n+1} \approx -\epsilon_n$. Deduce that the resulting algorithm gives cubic convergence with three function evaluations and hence has efficiency index $E = 3^{1/3} = 1.4422\ldots$. [3]

3. Another modification of the secant method can be derived by estimating $f'(x_n)$ in Newton's method by quadratic interpolation through the points x_n, x_{n-1}, x_{n-2}. Show that the resulting method can be written $x_{n+1} = x_n - f(x_n)/\omega$, where

$$\omega = f[x_n, x_{n-1}] + (x_n - x_{n-1})f[x_n, x_{n-1}, x_{n-2}].$$

4. The Muller–Traubś method uses three points to determine the coefficient of an interpolating parabola. The same points can also be interpolated by a rational function of the form

$$g(x) = \frac{x-a}{bx+c}.$$

   An iterative method is devised by taking x_{n+1} equal to the root a of $g(x) = 0$.

   (a) Show that this is equivalent to calculating x_{n+1} from the "modified secant formula"

$$x_{n+1} = x_n - f_n \frac{x_n - x_{n-2}}{f_n - \tilde{f}_{n-2}}, \qquad \tilde{f}_{n-2} = f_{n-2}\frac{f[x_n, x_{n-1}]}{f[x_{n-1}, x_{n-2}]}.$$

   Hint: Use a theorem in projective geometry, according to which the cross ratio of any four values of x is equal to the cross ratio of the corresponding values of $g(x)$ (see Householder [10, p. 159]). Hence

$$\frac{(0 - f_n)/(0 - f_{n-2})}{(y_{n-1} - f_n)/(y_{n-1} - f_{n-2})} = \frac{(x_{n+1} - x_n)/(x_{n+1} - x_{n-2})}{(x_{n-1} - x_n)/(x_{n-1} - x_{n-2})}.$$

[3] The resulting modified rule of false position is often called after its origin the **Illinois method**. It is due originally to the staff of the computer center at the University of Illinois in the early 1950's.

(b) Use the result in (a) to show that $x_{n-1} \in \operatorname{int}(x_{n-2}, x_n)$ if

$$\operatorname{sign}(y_n) = -\operatorname{sign}(y_{n-2}), \qquad \operatorname{sign}(y[x_n, x_{n-1}]) = \operatorname{sign}(y[x_{n-1}, x_{n-2}]).$$

5. The result in Problem 4 suggests that the Illinois method in Problem 2 is modified by taking

$$\beta = f[x_{n+1}, x_n]/f[x_n, x_{n-1}], \qquad \theta = \begin{cases} \beta, & \text{if } \beta > 0; \\ \frac{1}{2}, & \text{if } \beta \le 0. \end{cases}.$$

Implement this modified method. Compare it with the unmodified Illinois method and with the safeguarded secant algorithm. As test equations use the following:
(a) A curve with one inflection point on $[0, 1]$:
$f(x) = x^2 - (1 - x)^n$, $a = 25$, $b = 1$, $n = 2, 5, 10$.
(b) A family of curves which lie increasingly close to the x-axis for large n:
$f(x) = e^{-nx}(x - 1) + x^n$, $a = 0.25$, $b = 1$, $n = 5, 10, 15$.
(c) A family of curves with the y-axis asymptotic:
$f(x) = (nx - 1)/((n - 1)x)$, $a = 0.01$, $b = 1$, $n = 2, 5, 10$.

6.3 Methods Using Derivatives

6.3.1 Newton's method

When $f'(x)$ is available then **Newton's method**[4] (6.1.26) is usually the method of choice for solving an equation $f(x) = 0$. As shown already in Section 1.2.1, Newton's method is based on approximating the curve $y = f(x)$ by its tangent at the point $(x_n, f(x_n))$, where x_n is the current approximation to the root. Assuming that $f'(x_n) \ne 0$, the next approximation x_{n+1} is determined as the abscissa of the point of intersection of the tangent with the x-axis (see Figure 1.2.3). This is equivalent to replacing the equation $f(x) = 0$ by

$$T_n(x) = f(x_n) + (x - x_n)f'(x_n) = 0, \tag{6.3.1}$$

where $T_n(x)$ is obtained by truncating the Taylor expansion of $f(x)$ at x_n after the first two terms. Hence x_{n+1} is determined from

$$x_{n+1} = x_n + h_n, \quad h_n = -f(x_n)/f'(x_n). \tag{6.3.2}$$

Clearly Newton's method can be viewed as the limit of the secant method when the interpolation points coalesce,

[4]Newton's original method was more complicated and not very similar to what is now known as his method.

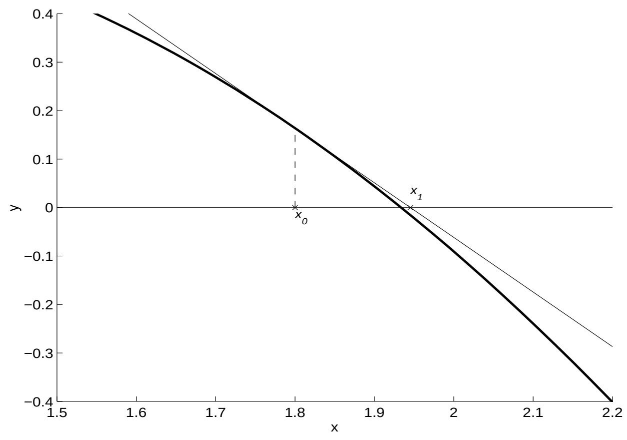

Figure 6.3.1. *Newton's method for the equation* $f(x) = (x/2)^2 - \sin x = 0$.

Example 6.3.1.

We want to compute the unique positive root of the equation $f(x) = (x/2)^2 - \sin x = 0$ (cf. Example 6.2.2) for which $f'(x) = x/2 - \cos x$. Starting from $x_0 = 1.8$ the first step of Newton's method is illustrated in Figure 6.3.1. The following Newton iterates are given in the table below (correct digits in x_n shown in bold):

n	x_n	$f(x_n)$	$f'(x_n)$	h_n
0	**1**.8	−0.163847 630878	1.127202 094693	−0.145357 812631
1	**1.9**45357 812631	0.015436 106659	1.338543 359427	0.011532 018406
2	**1.933**825 794225	0.000095 223283	1.322020 778469	0.000072 028582
3	**1.933753 76**5643	0.000000 003722	1.3219174 29113	0.000000 002816
4	**1.933753 762827**			

The number of correct digits approximately double in each iteration until the limiting precision is reached. Although the initial approximation is not very good, already x_4 is correct to twelve decimals!

If the iterations are broken off when $|h_n| < \delta$ it can be shown (see the error analysis below) that the truncation error is less than δ, provided that $|Kh_n| \le 1/2$, where K is an upper bound for $|f''/f'|$ in the neighborhood of the root. This restriction is seldom of practical importance. However, rounding errors made in computing h_n must also be taken into account.

Note that when we approach the root, the *relative* precision in the computed values of $f(x_n)$ usually becomes less and less. Since $f'(x_n)$ is only used for computing h_n it need not be computed to much greater *relative precision* than $f(x_n)$, In the above example we could have used $f'(x_2)$ instead of $f'(x_n)$ instead for $n > 2$ without much affecting the convergence. Such a simplification is of great importance when Newton's method is used on *systems* of nonlinear equations; see Volume II, Section 11.2.4.

We first consider the **local** convergence of Newton's method, that is the convergence in a neighborhood of a root α. Assume that $f'(x)$ is continuous in a neighborhood of α and that $f'(\alpha) \neq 0$. Expanding f in a Taylor series about x_0 we get

$$0 = f(\alpha) = f(x_n) + (\alpha - x_n)f'(\xi_n), \quad \xi_n \in \mathrm{int}(x_n, \alpha).$$

We let $\epsilon_n = x_n - \alpha$ denote the error in the approximation x_n. Subtracting (6.3.1) with $x = x_{n+1}$ and using $x_{n+1} - x_n = \epsilon_{n+1} - \epsilon_n$, we have

$$-\epsilon_n f'(\xi_n) = (\epsilon_{n+1} - \epsilon_n)f'(x_n),$$

or after dividing by $f'(x_n)$

$$\epsilon_{n+1} = \left(1 - \frac{f'(\xi_n)}{f'(x_n)}\right)\epsilon_n, \quad n = 0, 1, 2, \dots.$$

Hence, if x_0 is sufficiently close to α, then $\lim_{n\to\infty} x_n = \alpha$. In other words α is a point of attraction of the Newton iteration and *Newton's method always converges (to a simple root) from a sufficiently good starting approximation.*

For convergence f need only have *one* continuous derivative. To get a more precise relation between ϵ_{n+1} and ϵ_n we assume in what follows that f has *two* continuous derivatives.

Theorem 6.3.1. *Assume that α is a simple root of the equation $f(x) = 0$, i.e., $f'(\alpha) \neq 0$. If f' exists and is continuous in a neighborhood of α, then the convergence order of Newton's method is at least equal to two.*

Proof. A Taylor expansion of f yields

$$0 = f(\alpha) = f(x_n) + (\alpha - x_n)f'(x_n) + \frac{1}{2}(\alpha - x_n)^2 f''(\zeta_n), \quad \zeta_n \in \mathrm{int}(x_n, \alpha).$$

Subtracting $f(x_n) + (x_{n+1} - x_n)f'(x_n) = 0$ and solving for $\epsilon_{n+1} = x_{n+1} - \alpha$ we get

$$\epsilon_{n+1} = \frac{1}{2}\epsilon_n^2 \frac{f''(\zeta_n)}{f'(x_n)}, \quad \zeta_n \in \mathrm{int}(x_n, \alpha). \tag{6.3.3}$$

Provided that $f'(\alpha) \neq 0$, it follows that (6.1.22) is satisfied with $p = 2$ and the asymptotic error constant is

$$C = \frac{1}{2}\left|\frac{f''(\alpha)}{f'(\alpha)}\right|. \tag{6.3.4}$$

If $f''(\alpha) \neq 0$, then $C > 0$ and the rate of convergence is quadratic. $\square$

Note that *the relation* (6.3.3) *between the errors only holds as long as the round-off errors in the calculations can be ignored.* As pointed out in Section 6.1.3, the limiting factor for the accuracy, which can be achieved in calculating the root, is always limited by the accuracy of the computed values of $f(x)$.

So far we have assumed that α is a simple root. Suppose now that α is a root of multiplicity $q > 1$. Then by Taylor's formula we have (cf. (6.1.9))

$$f'(x) = \frac{1}{(q-1)!}(x-\alpha)^{q-1} f^{(q)}(\xi'), \quad \xi' \in \operatorname{int}(x, \alpha).$$

It follows that if x_n is close to α, then the Newton correction will satisfy

$$h_n = \frac{f(x_n)}{f'(x_n)} \approx \frac{1}{q}(x_n - \alpha) = \epsilon_n/q.$$

For the corresponding errors we have

$$\epsilon_{n+1} = \epsilon_n - \epsilon_n/q = (1 - 1/q)\epsilon_n,$$

which shows that for a root of multiplicity $q > 1$ Newton's method only converges *linearly* with rate $C = 1 - 1/q$. (The same is true of other methods which have quadratic or higher rate of convergence for simple roots.) For $q > 2$ this is much slower even than for the bisection method! Note also that when $x_n \to \alpha$ both $f(x_n) \to 0$ and $f'(x_n) \to 0$. Therefore rounding errors may seriously affect the Newton correction when evaluated close to α, and some safeguarding is essential; see Section 6.3.2.

When the multiplicity p of a root is known a priori the modified Newton's method

$$x_{n+1} = x_n - p\frac{f(x_n)}{f'(x_n)}, \tag{6.3.5}$$

is easily shown to have quadratic convergence. For a root of unknown multiplicity we can use the following observation. From (6.1.9) it follows that the equation $u(x) = 0$, where

$$u(x) = f(x)/f'(x), \tag{6.3.6}$$

always has *a simple root at* $x = \alpha$. Hence if Newton's method is applied to this equation it will retain its quadratic rate of convergence independent of the multiplicity of α as a root to $f(x) = 0$. The iteration becomes

$$x_{n+1} = x_n - \frac{f(x_n)f'(x_n)}{(f'(x_n))^2 - f(x_n)f''(x_n)}, \tag{6.3.7}$$

and thus requires the evaluation also of $f''(x_n)$.

Newton's method applied to the equation $f(x) = x^p - c = 0$ can be used to compute $c^{1/p}$, $p = \pm 1, \pm 2, \ldots$. The sequence $x_1, x_2, x_3, \ldots$, is then computed recursively from

$$x_{n+1} = x_n - \frac{x_n^p - c}{p x_n^{p-1}},$$

which can be written as

$$x_{n+1} = \frac{1}{p}\left((p-1)x_n + \frac{c}{x_n^{p-1}}\right) = \frac{x_n}{(-p)}[(1-p) - c x_n^{-p}]. \tag{6.3.8}$$

It is convenient to use the first expression in (6.3.8) when $p > 0$ and the second when $p < 0$. This iteration formula is often used for calculating, e.g., $\sqrt{c}$, $\sqrt[3]{c}$, and $1/\sqrt{c}$, corresponding to $p = 2, 3$, and -2 respectively. Note also that $1/c$, $(p = -1)$ can be computed by the iteration

$$x_{n+1} = x_n + x_n(1 - cx_n) = x_n(2 - cx_n),$$

using only multiplications and addition. In some early computers, which lacked built-in division, this iteration was used to implement division, i.e. b/c was computed as $b \times (1/c)$.

Example 6.3.2.

We want to construct an algorithm based on Newton's method for efficiently computing the square root of any given floating point number a. If we first shift the mantissa so that the exponent becomes even, $a = c \cdot 2^{2e}$, and $1/2 \le c < 2$, then

$$\sqrt{a} = \sqrt{c} \cdot 2^e.$$

We need only consider the reduced range $1/2 \le c \le 1$ since for $1 < c \le 2$ we can compute $\sqrt{1/c}$ and invert.

To find an initial approximation x_0 to start the Newton iterations when $1/2 \le c < 1$, we can use linear interpolation of $x = \sqrt{c}$ between the endpoints $1/2, 1$, giving

$$x_0(c) = \sqrt{2}(1 - c) + 2(c - 1/2)$$

($\sqrt{2}$ is precomputed). The iteration then proceeds with

$$x_{n+1} = \frac{1}{2}\Big(x_n + \frac{c}{x_n}\Big), \quad n = 0, 1, 2, \dots. \tag{6.3.9}$$

For $c = 3/4$ ($\sqrt{c} = 0.86602540378444$) the result is $x_0 = (\sqrt{2} + 2)/4$ and (correct digits in boldface)

$$\begin{aligned} x_0 &= \mathbf{0.8}5355339059327, & x_1 &= \mathbf{0.866}11652351682, \\ x_2 &= \mathbf{0.86602540}857756, & x_3 &= \mathbf{0.86602540378444}, \end{aligned}$$

Three iterations suffice to give full IEEE double precision accuracy and the quadratic rate of convergence is apparent.

6.3.2 Global Convergence of Newton's Method

It is easy to construct examples where Newton's method converges very slowly or not at all. Recall, e.g., that for a root of multiplicity $q > 1$ convergence is linear with rate $1 - 1/q$. For $q = 20$ it will take 45 iterations to gain one more decimal digit.

Example 6.3.3.

The equation $f(x) = \sin x = 0$, has exactly one root $\alpha = 0$ in the interval $|x| < \pi/2$. Newton's method becomes

$$x_{n+1} = x_n - \tan x_n, \quad n = 0, 1, 2, \ldots.$$

If we choose the initial value $x_0 = x^*$ such that $\tan x^* = 2x^*$, then $x_1 = -x_0$, $x_2 = -x_1 = x_0$. Hence the successive approximations show a cyclic behavior!

Newton's method will converge for any starting value such that $|x_0| < x^*$. The critical value can be shown to be $x^* = 1.16556\ldots$.

In some simple cases the global convergence of Newton's method may be easy to verify. Two examples are given in the following theorems.

Theorem 6.3.2. *Suppose that $f'(x)f''(x) \neq 0$ in an interval $[a, b]$, where $f''(x)$ is continuous and $f(a)f(b) < 0$. Then if*

$$\left|\frac{f(a)}{f'(a)}\right| < b - a, \qquad \left|\frac{f(b)}{f'(b)}\right| < b - a,$$

Newton's method converges from an arbitrary $x_0 \in [a, b]$.

Proof. The theorem follows easily by inspecting Figure 6.4.2. □

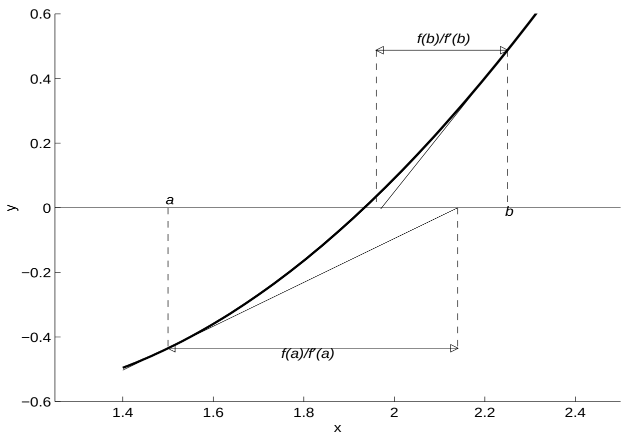

Figure 6.3.2. *A situation where Newton's method converges from any $x_0 \in [a, b]$.*

Lemma 6.3.3. *Let $[a, b]$ be an interval such that $f(a)f(b) < 0$. Assume that the so-called* **Fourier conditions** *are satisfied, i.e., $f'(x)f''(x) \neq 0$ in $[a, b]$, with $f''(x)$ continuous and $f(x_0)f''(x_0) > 0$, for $x_0 \in [a, b]$. Then the sequence $\{x_0, x_1, x_2, \ldots\}$ generated by Newton's method converges monotonically to a root $\alpha \in [a, b]$.*

Proof. We can assume that $f''(x) > 0$; otherwise consider the equation $-f(x) = 0$. Assume first that $f'(x) < 0$ in the interval. Since by assumption $f(x_0) \geq 0$, this corresponds to the situation in Figure 6.4.2, with $b = x_0 > \alpha$. Clearly it holds that $x_1 > x_0$ and since the curve lies to the left of the tangent in x_0 we also have $x_1 > \alpha$. The case when $f'(x) < 0$ can be treated similarly. The theorem now follows by induction. $\square$

Newton's method can be safeguarded by taking a bisection step whenever a Newton step "fails" in some sense. Assume that initially $a < b$ and $f(a)f(b) < 0$ and x is either equal to a or b. At each step a new approximation x' is computed and a, b are updated to a', b' as follows:

- If the Newton iterate $x' = x - f(x)/f'(x)$ lies in (a, b), then accept x'; otherwise take a bisection step, i.e. set $x' = (a + b)/2$.
- Set either $a' = x$, $b' = b$ or $a' = a$, $b' = x$, where the choice is made so that $f(a')f(b') \leq 0$.

This ensures that at each step the interval $[a', b']$ contains a root.

When checking if $z \in (a, b)$, it is important to avoid division by $f'(x)$, since this may cause overflow or division by zero. Hence, we note $z \in (a, b)$ if and only if

$$b - z = b - x + f(x)/f'(x) > 0 \quad \text{and} \quad z - a = x - a - f(x)/f'(x) \geq 0.$$

If $f'(x) > 0$ these two inequalities are equivalent to

$$(b - x)f'(x) > -f(x) \quad \text{and} \quad (x - a)f'(x) > f(x).$$

The case when $f'(x) < 0$ is analyzed similarly, giving

$$(b - x)f'(x) < -f(x) \quad \text{and} \quad (x - a)f'(x) < f(x).$$

In either case only one of the inequalities will be nontrivial depending on whether $f(x) > 0$ or not.

6.3.3 Newton Method for Complex Roots

Newton's method is based on approximating f with the linear part of its Taylor expansion. Taylor's theorem is valid for a complex function $f(z)$ around a point of analyticity a (see Sec. 3.1.3). Hence Newton's method applies also to an equation $f(z) = 0$, where $f(z)$ is a complex function, analytic in a neighborhood of a root α. An important example is when f is a polynomial; see Sec. 6.5.

Let $z = x + iy$, $f(z) = u(x, y) + iv(x, y)$, and consider the absolute value of $f(z)$

$$\phi(x, y) = |f(x + iy)| = \sqrt{u(x, y)^2 + v(x, y)^2}.$$

This is a differentiable function as a function of (x, y), except where $f(z) = 0$. The gradient of $\phi(x, y)$ is

$$\operatorname{grad}\phi = (\phi_x, \phi_y) = \frac{1}{\phi}\,(uu_x + vv_x, uu_y + vv_y) \tag{6.3.10}$$

where $u_x = \partial u/\partial x$, $u_y = \partial u/\partial y$, etc. Using the Cauchy–Riemann equations $u_x = v_y$, $u_y = -v_x$ we calculate (see Henrici [9, §6.1.4]).

$$\frac{f(z)}{f'(z)} = \frac{u + iv}{u_x + iv_x} = \frac{(uu_x + vv_x) + i(uu_y + vv_y)}{u_x^2 + v_x^2},$$

A comparison with (6.3.10) shows that the Newton step

$$z_{k+1} - z_k = -f(z_k)/f'(z_k), \tag{6.3.11}$$

is in the direction of the negative gradient of $|f(z_k)|$, i.e., in the direction of strongest decrease of $|f(z)|$.

The geometry of the complex Newton iteration is considered in [29].

Theorem 6.3.4.

Let the function $f(z) = f(x + iy)$ be analytic and $z_k = x_k + iy_k$ be a point such that $f(z_k)f'(z_k) \neq 0$. Let z_{k+1} be the next iterate of Newton's method (6.3.11). *Then $z_{k+1} - z_k$ is in the direction of the negative gradient of $\phi(x, y) = |f(x + iy)|$ and therefore orthogonal to the level set of $|f|$ at (x_k, y_k). If T_k is the tangent plane of $\phi(x, y)$ at (x_k, y_k) and L_k is the line of intersection of T_k with the xy-plane then (x_{k+1}, y_{k+1}) is the point on L_k closest to (x_k, y_k).*

Proof. See [29]. □

Newton's method is very efficient if started from an initial approximation sufficiently close to a simple zero. If this is not the case Newton's method may converge slowly or even diverge. In general, there is no guarantee that z_{n+1} is closer to the root than z_n, and if $f'(z_n) = 0$ the next iterate is not even defined.

The following important theorem gives rigorous sufficient conditions for the global convergence of Newton's method. For generality we formulate it for complex zeros of a complex valued function $f(z)$.

Theorem 6.3.5.

Let $f(z)$ be a complex function of a complex variable. Let $f(z_0)f'(z_0) \neq 0$ and set $h_0 = -f(z_0)/f'(z_0)$, $x_1 = x_0 + h_0$. Assume that $f(z)$ is twice continuously differentiable in the disk $K_0 : |z - z_1| \leq |h_0|$, and that

$$2\,|h_0|M_2 \leq |f'(z_0)|, \quad M_2 = \max_{z \in K_0} |f''(x)|. \tag{6.3.12}$$

Let z_k be generated by Newton's method

$$z_{k+1} = z_k - \frac{f(z_k)}{f'(z_k)}, \quad k = 1, 2, \ldots.$$

Then $z_k \in K_0$ and we have $\lim_{k\to\infty} z_k = \zeta$, where ζ is the only zero of $f(z)$ in K_0. Unless ζ lies on the boundary of K_0, ζ is a simple zero. Further we have the relations

$$|\zeta - z_{k+1}| \leq \frac{M_2}{2|f'(z_k)|}|z_k - z_{k-1}|^2, \quad k = 1, 2, \ldots. \tag{6.3.13}$$

In case of a real function $f(x)$ of a real variable the disk K_0 can be replaced by the closed interval $K_0 = \mathrm{int}\,[x_0, x_0 + 2h_0]$.

Proof. See Ostrowski [19, Theorem 7.2] □

A generalization of this result, Theorem 11.1.7 (the Newton–Kantorovich Theorem), for the multivariate Newton method, will be proved in Volume II.

Since the Newton step is in the direction of the negative gradient of $|f(z)|$ at $z = z_k$, it will necessarily give a decrease in $|f(z_k)|$ if a short enough step in this direction is taken. A modified Newton method based on the descent property and switching to standard Newton when the condition (6.3.12) is satisfied, will be described in Sec. 6.5.6.

6.3.4 An Interval Newton Method

.

Sometimes it is desired to compute a tiny interval that is guaranteed to enclose a real simple root x^* of $f(x)$, even when rounding errors are taken into account. This can be done using an adaptation of Newton's method to interval arithmetic method due to Moore [17].

Suppose that the function $f(x)$ is continuously differentiable. Using the notation in Sec. 2.3.7, let $f'([x_0])$ denote an interval containing $f'(x)$ for all x in a finite interval $[x] := [a, b]$. Define the Newton operator on $N[x]$ by

$$N([x]) := m - \frac{f(m)}{f'([x])}. \qquad (6.3.14)$$

where $m = \mathrm{mid}\,([x]) = \frac{1}{2}(a + b)$.

Theorem 6.3.6.

If $\alpha \in [x]$ is a zero of $f(x)$, then $\alpha \in N([x])$. If $N([x]) \subseteq [x]$, then $f(x)$ has one and only one zero in $N([x])$.

Proof. Suppose α is a zero of $f(x)$ in $[x]$. If $0 \in f'([x])$ then $N([x]) = [-\infty, \infty]$. Otherwise, by the mean value theorem

$$0 = f(\alpha) = f(m) + f'(\xi)(\alpha - m), \quad \xi \in \mathrm{int}\,[\alpha, m] \subseteq [x].$$

This implies that $\alpha = m - f(m)/f'(\xi) \subseteq N([x])$, which proves the first statement.

If $N([x]) \subseteq [x]$ then $f'([x]) \neq 0$ on $[x]$. Then by the mean value theory there are ξ_1 and ξ_2 in $[x]$ such that

$$\begin{aligned}(m - f(m)/f'(\xi_1)) - a &= -f(a)/f'(\xi_1),\\ b - (m - f(m)/f'(\xi_2)) &= f(b)/f'(\xi_2).\end{aligned}$$

Because $N([x]) \subseteq [a, b]$, the product of the left sides is positive. But since $f'(\xi_1)$ and $f'(\xi_2)$ have the same sign this means that $f(a)f(b) < 0$ and f has therefore a zero in $[x]$.

Finally, there cannot be two or more zeros in $[x]$, because then we would have $f'(c) = 0$ for some $c \in [x]$. □

In the interval Newton method, a starting interval $[x_0]$ is chosen, and we compute for $k = 0, 1, 2, \dots$ a sequence of intervals $[x_{k+1}]$ given by

$$N([x_k]) = \operatorname{mid}([x_k]) - \frac{f(\operatorname{mid}[x_k])}{f'([x_k])}.$$

If $N([x_k]) \subset [x_k]$ we set $[x_{k+1}] = N([x_k]) \cap [x_k]$. Otherwise, if $N([x_k]) \cap [x_k]$ is empty, we know that $[x_k]$ does not contain a root and stop. In neither condition holds we stop, subdivide the initial interval and start again. It can be shown that if $[x_0]$ does not contain a root then after a finite number of steps the iteration will stop with an empty interval.

If we are close enough to a zero, then the length of the intervals $[x_k]$ will converge quadratically to zero, just as the standard Newton method.

Example 6.3.4.

Take $f(x) = x^2 - 2$ and $[x_0] = [1, 2]$. Using interval Newton method

$$N([x_k]) = \operatorname{mid}([x_k]) - \frac{(\operatorname{mid}[x_k])^2 - 2}{2\,[x_k]}, \quad [x_{k+1}] = N([x_k]) \cap [x_k].$$

we obtain the sequence of intervals

$$[x_1] = N([x_0]) = 1.5 - \frac{2.25 - 2}{2[1,2]} = [22/16, 23/16] = [1.375, 1.4375],$$

$$[x_2] = N([x_1]) = \frac{45}{32} - \frac{(45/32)^2 - 2}{2[22/16, 23/16]} = \frac{45}{32} - \frac{(45)^2 - 2(32)^2}{128\,[22, 23]} \subset [1.41406, 1.41442].$$

The quadratic convergence of the radius of the intervals is evident:

$$0.5, 0.03125, 0.00036, \dots.$$

The interval Newton method, is well suited to determine *all zeros in a given interval.* Divide the given interval into subintervals and for each subinterval $[x]$ check whether the condition $N([x]) \subseteq [x]$ in Theorem 6.3.6 holds. If this is the case, we continue the interval Newton iterations, and if we are close enough the iterations converge towards a root. If the condition is not satisfied but $N([x]) \cap [x]$ is empty then there is no zero in the subinterval and this can be discarded. If the condition fails but $N([x]) \cap [x]$ is not empty, then subdivide the interval and try again. The calculations can be organized so that we have a queue of intervals waiting to be precessed. Intervals may be added or removed form the queue. When the queue is empty we are done.

The above procedure may not always work. Its performance will depend among other things on the sharpness of the inclusion of the derivative $f'([x])$. Things will go wrong, e.g., in case of multiple roots where $N([x]) = [-\infty, \infty]$.

6.3.5 Higher Order Methods

Newton's method has quadratic convergence, which means that the number of significant digits approximately doubles in each iteration. Although there is rarely any practical need for methods of higher order of convergence such methods may be useful in special applications.

For the following discussion we assume that α is a simple zero of f and that f has a sufficient number of continuous derivatives in a neighborhood of α. We first briefly review some famous methods with cubic convergence for simple roots of $f(x) = 0$.

Newton's method was derived by approximating the function $f(x)$ with its linear Taylor approximation. Higher order iteration methods can be constructed by including more terms from the Taylor expansion. The quadratic Taylor approximation of the equation $f(x_n + h) = 0$ is

$$f(x_n) + hf'(x_n) + \frac{h^2}{2} f''(x_n) = 0, \quad h = x - x_n. \tag{6.3.15}$$

Assuming that $f'(x_n)^2 \geq 2f(x_n)f''(x_n)$, the solutions of this quadratic equation are real and equal to

$$h_n = -\frac{f'(x_n)}{f''(x_n)} \left(1 \pm \sqrt{1 - 2\frac{f(x_n)f''(x_n)}{(f'(x_n))^2}} \right).$$

Rearranging and taking the solution of smallest absolute value we get

$$x_{n+1} = x_n - u(x_n) \cdot \frac{2}{1 + \sqrt{1 - 2t(x_n)}}, \tag{6.3.16}$$

where we have introduced the notation

$$u(x) = \frac{f(x)}{f'(x)}, \qquad t(x) = u(x)\frac{f''(x)}{f'(x)}. \tag{6.3.17}$$

The iteration (6.3.16) is **Euler's iteration method**.

Assuming that $|t(x_n)| \ll 1$ and using the approximation

$$\frac{2}{1 + \sqrt{1 - 2t(x_n)}} \approx 1 + \tfrac{1}{2}t(x_n), \tag{6.3.18}$$

valid for $|t| \ll 1$, we obtain another third order iteration method usually also attributed to Euler

$$x_{n+1} = x_n - u(x_n)\big(1 + \tfrac{1}{2}t(x_n)\big). \tag{6.3.19}$$

A different method of cubic convergence is obtained by using a rational approximation of (6.3.18)

$$x_{n+1} = x_n - u(x_n) \cdot \frac{1}{1 - \frac{1}{2}t(x_n)} = x_n - \frac{f(x_k)}{f'(x_k) - \dfrac{f''(x_k)}{2f'(x_k)} f(x_k)}. \tag{6.3.20}$$

This is **Halley's**[5] iteration method [8], which has the distinction of being the most frequently rediscovered iteration method. Halley's method has a simple geometric interpretation. Consider a hyperbola

$$h(x) = b + \frac{a}{x-c},$$

where a, b, c are determined so that $h(x)$ is osculatory to f at $x = x_n$, i.e., is tangent to the curve at this point and has the same curvature there. Then x_{n+1} is the intersection of this hyperbola with the x-axis.

The methods (6.3.19) and (6.3.20) correspond to the (1,0) and (0,1) Padé approximations of Euler's method. We now show that both are of third order for simple zeros. Following Gander [6] we consider an iteration function of the general form

$$\phi(x) = x - u(x)H(t(x)), \tag{6.3.21}$$

where $u(x)$ and $t(x)$ are defined by (6.3.17). Differentiating (6.3.21) and using $u'(x) = 1 - t(x)$ we get

$$\phi'(x) = 1 - (1 - t(x))H(t) - u(x)H'(t)t'(x).$$

Since $u(\alpha) = t(\alpha) = 0$ it follows that $\phi'(\alpha) = 1 - H(0)$. Hence if $H(0) = 1$ then $\phi'(\alpha) = 0$ and the iteration function (6.3.21) is at least of second order. Differentiating once more and putting $x = \alpha$ we get

$$\phi''(\alpha) = t'(\alpha)H(0) - 2u'(\alpha)H'(0)t'(\alpha) = t'(\alpha)(H(0) - 2H'(0)).$$

Hence $\phi'(\alpha) = \phi''(\alpha) = 0$ and the method (6.3.21) at least of third order if the conditions

$$H(0) = 1, \qquad H'(0) = 1/2 \tag{6.3.22}$$

are satisfied. For Euler's and Halley's method we have

$$H(t) = 2\left(1 + \sqrt{1-2t}\right)^{-1}, \qquad H(t) = (1 - \tfrac{1}{2}t)^{-1},$$

respectively, and both these methods satisfy the conditions in (6.3.22). (Verify this!)

Example 6.3.5.

Using (6.3.20) a short calculation shows that Halley's method for solving the equation $f(x) = x^2 - c = 0$ can be written

$$x_{n+1} = x_n - 2x_n \frac{x_n - c/x_n}{3x_n + c/x_n}, \quad n = 0, 1, 2, \ldots. \tag{6.3.23}$$

(see Problem 10). For $c = 3/4$ we and using the initial approximation $x_0 = (\sqrt{2} + 2)/4$ obtained by linear interpolation in Example 6.3.2 we obtain the following result (correct digits in boldface)

$$x_0 = \mathbf{0.8}5355339059327, \quad x_1 = \mathbf{0.866024}74293290, \quad x_2 = \mathbf{0.86602540378444}.$$

[5] Edmund Halley (1656–1742), an English astronomer, who predicted the periodic reappearance (c:a 75 years) of a comet named after him.

Already two iterations give a result correct to 14 digits. Compared to Newton's method we have gained one iteration, but each iteration is more costly.

It is not difficult to construct iteration methods of arbitrarily high order for solving $f(x) = 0$. It can be shown ([25, Theorem 5.3]) that any iteration function of order p must depend explicitly on the first $p-1$ derivatives of f. Consider the Taylor expansion at x_n

$$0 = f(x_n + h) = f(x_n) + hf'(x_n) + \sum_{k=2}^{p-1} \frac{h^k}{k!} f^{(k)}(x_n) + O(h^p). \tag{6.3.24}$$

Neglecting the $O(h^p)$-term this is a polynomial equation of degree $p-1$ in h. Assuming that $f'(x_n) \neq 0$ we could solve this by the fixed point iteration $h_i = F(h_{i-1})$, where

$$F(h) = -\frac{f(x_n) + \sum_{k=2}^{p-1} h^k f^{(k)}(x_n)/k!}{f'(x_n)},$$

taking h_0 to be the Newton correction.

To get an explicit method of order p we write

$$\frac{f(x_n)}{f'(x_n)} \equiv u = -h - \sum_{k=2}^{p-1} a_k h^k, \tag{6.3.25}$$

where

$$a_k = \frac{f^{(k)}(x_n)}{k! f'(x_n)}, \quad k \geq 2. \tag{6.3.26}$$

This can be interpreted as a formal power series in h (cf. Sec. 3.1.5). Reversing this series we can express h as a formal power series in u

$$h = -u - \sum_{k=2}^{p-1} c_k u^k + \cdots, \tag{6.3.27}$$

$$c_2 = a_2, \qquad c_3 = 2a_2^2 - a_3, \qquad c_4 = 5a_2^3 - 5a_2 a_3 + a_4, \tag{6.3.28}$$
$$c_5 = 14a_2^4 - 21a_2^2 a_3 + 6a_2 a_4 + 3a_3^2 - a_5, \ldots.$$

More coefficients can easily be determined; see Problem 3.1.12. This leads to the **Schröder family** of iteration methods

$$x_{n+1} = x_n - u(x_n) - \sum_{k=2}^{p-1} c_k(x_n) u^k(x_n), \tag{6.3.29}$$

(E. Schröder [22]). If f is analytic these can be shown to have convergence order p for simple roots. For a proof see Henrici [9, p. 529].

Setting $p = 2$ gives Newton's method and for $p = 3$ we get the third order method of Euler (6.3.19). The family (6.3.29) of high order methods makes use of polynomials in u. As for the case $p = 3$, variants of these using rational expressions in u can be derived from Padé approximants of these polynomials; see Traub [25, Sec. 5.2]. There are indications that methods which use rational approximations with about equal degree of nominator and denominator are best. For example, for $p = 4$, the rational approximation

$$1 + c_2 u + c_3 u^2 = \frac{c_2 + (c_2^2 - c_3)u}{c_2 - c_3 u} + \mathcal{O}(u^3)$$

can be used to derive an alternative method of order 4.

We now introduce a rational family of iteration methods of arbitrary order, which is very convenient to use. First note that Halley's method can also be derived as follows. Starting from (6.3.15) we get

$$h_n = -f(x_n) \Big/ \Big(f'(x_n) + \frac{h_n}{2} f''(x_n) \Big).$$

Replacing h_n in the denominator by the Newton correction $-f(x_n)/f'(x_n)$, does not change the order of the method and leads to (6.3.20). We note that this can be written $x_{n+1} = x_n + B_2(x_n)$, where

$$B_2(x) = -f(x) \, \frac{f'(x)}{\det \begin{pmatrix} f'(x) & \frac{f''(x)}{2!} \\ f(x) & f'(x) \end{pmatrix}}.$$

This method belongs to a rational family of iteration function of arbitrary order, which we now give. Set $D_p(x) = \det(F_p)$, where

$$F_p(x) = \begin{pmatrix} f'(x) & \frac{f''(x)}{2!} & \dots & \frac{f^{(p-1)}(x)}{(p-1)!} & \frac{f^{(p)}(x)}{(p)!} \\ f(x) & f'(x) & \ddots & \ddots & \frac{f^{(p-1)}(x)}{(p-1)!} \\ 0 & f(x) & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \frac{f''(x)}{2!} \\ 0 & 0 & \dots & f(x) & f'(x) \end{pmatrix} \in \mathbf{R}^{p \times p}, \qquad (6.3.30)$$

is a Toeplitz upper Hessenberg matrix defined with respect to the normalized derivatives of $f(x)$. (Recall that a square matrix is called Toeplitz if its elements are identical along each diagonal.) The iteration

$$x_{n+1} = x_n + B_p(x_n), \quad B_p(x) = -f(x) \frac{\det(F_{p-2}(x))}{\det(F_{p-1}(x))} \qquad (6.3.31)$$

can be shown to be of order p for simple roots.

The determinant formula (6.3.31) is attractive since it leads to a simple implementation. To evaluate $\det(F_p(x))$ we use Gaussian elimination (without pivoting) to compute the LU factorization

$$F_p(x_n) = L_p U_p, \qquad \operatorname{diag}(U_p) = (u_{11}, u_{22}, \dots, u_{pp}).$$

where L_p is unit lower triangular and U_p upper triangular; see Sec. 1.3.4. Then $\det(F_p) = u_{11}u_{22}\dots u_{pp}$. Hence the ratio $\det(F_p(x))/\det(F_{p-1}(x))$ equals the *last diagonal element* u_{pp} in U_p. It follows that

$$x_{n+1} = x_n - f(x_n)/u_{kk}(x_n), \quad k = 1:p, \tag{6.3.32}$$

gives the result of a sequence of iteration formulas of order $2:p+1$.

Note that the Gaussian elimination is simplified by the fact that $F_p(x_n)$ is Hessenberg. Only the $p-1$ subdiagonal elements need to be eliminated, which requires $\frac{1}{2}p^2$ flops. Further, it is possible to implement the computation of diag (U_p) using only two row vectors; see Problem 14.

The main drawback to the use of higher order iteration methods is the computational cost of evaluating higher derivatives of f. However, if f satisfies a differential equation higher order derivatives can be calculated by differentiating the differential equation.

Example 6.3.6.

Methods using high order derivatives are useful in particular when seeking zeros of a function satisfying a (usually second order) differential equation. The Bessel functions of the first kind $\mathcal{J}_\nu(x)$ satisfies the differential equation

$$x^2y'' + xy' + (x^2 - \nu^2)y = 0.$$

The smallest zero ξ of $\mathcal{J}_0(x)$ is close to $x_0 = 2.40$, and

$$\mathcal{J}_0(x_0) = 0.00250\,76832\,9724, \qquad \mathcal{J}_0'(x_0) = -\mathcal{J}_1(x) = -0.52018\,52681\,8193.$$

Differentiating the differential equation for $\mathcal{J}_0(x)$ we get

$$xy^{(k+1)} + ky^{(k)} + xy^{(k-1)} + (k-1)y = 0, \quad k \geq 1,$$

which gives a recursion for computing higher derivatives. Taking $p = 5$

$$y''(x_0)/2! = 0.10711\,80892\,2261, \qquad y'''(x_0)/3! = 0.05676\,83752\,3951,$$
$$y^{iv}(x_0)/4! = -0.00860\,46362\,1903.$$

Forming the Toeplitz matrix F_4 and computing the diagonal elements of U in the LU factorization, we obtain using (6.3.32) the following sequence of approximations to ξ:

2.40482 07503 **2.40482 55**406 **2.40482 55576** 7553, **2.40482 55576 9573**

(correct digits shown in boldface).

Review Questions

1. (a) Under what assumptions is convergence of Newton's method quadratic?

   (b) Device an example where Newton's method diverges, even though the equation has real roots.
2. Describe an iteration for the division-free computation of the reciprocal of a positive number c. Determine the largest set of starting values x_0 such that the iterates converge to $1/c$.
3. The equation $f(x) = \sin x = 0$ has one trivial root $x = 0$ in the interval $(-\pi/2, \pi/2)$. Show that for an initial approximation x_0 chosen so that $\tan x_0 = 2x_0$ Newton's method cycles, and $x_{2k} = x_0$ for all $k \geq 0$!
4. (a) Assume that f is continuously differentiable in a neighborhood of a double root α of the equation $f(x) = 0$. Describe how the equation can be converted to one with a simple root α.

   (b) Discuss the case when $f(x) = 0$ has two distinct roots which *nearly* coincide.

Problems and Computer Exercises

1. (a) Compute $\epsilon_{n+1}/\epsilon_n^2$ for $n = 0, 1, 2$, and the limit, as $n \to \infty$, in Example 6.3.1.

   (b) Treat the equation in Example 6.3.1 using $f'(x_2)$ as a fixed approximation to $f'(x_n)$ for $n > 2$. Compare the convergence of this simplified method with the true Newton method..
2. The equation $x^3 - 2x - 5 = 0$ is of historical interest because it was the one used by Wallis[6] to present Newton's method to the French Academy. Determine the roots of this equation.

   Hint: It has one real and two complex roots.
3. Use Newton's method to determine the positive root of the equation to six correct decimals: (a) $x = 1 - e^{-2x}$; (b) $x \ln x - 1 = 0$
4. Determine the unique positive real root of the equation $x^q - x - 1 = 0$ for $q = 2 : 8$.
5. (a) Consider the Newton iteration used in Example 6.3.2 for computing square root. Show that the iterations satisfy

$$x_{n+1} - \sqrt{c} = \frac{1}{2x_n}(x_n - \sqrt{c})^2.$$

   Use this relation to show that, for all $x_0 > 0$, convergence is monotone $x_1 \geq x_2 \geq x_3 \geq \cdots \geq \sqrt{c}$ and that $\lim_{n\to\infty} = \sqrt{c}$ (compare Figure 1.2.5).

   (b) In Example 6.3.2 Newton's method was used to compute $\sqrt{c}$ for $1/2 \leq c \leq 1$. Determine the maximum error of the linear initial approximation used there. Then use the expression for the error in (a) to determine the number of iterations that suffices to give $\sqrt{c}$ with an error less than 10^{-14} for *all* c in

[6] John Wallis (1616–1703) the most influential English mathematician before Newton.

$[1/2, 1]$ using this initial approximation. Show that the influence of rounding errors is negligible.

6. Determine p, q and r so that the order of the iterative method

$$x_{n+1} = px_n + qc/x_n^2 + rc^2/x_n^5$$

for computing $\sqrt[3]{c}$ becomes as high as possible. For this choice of p, q and r, give a relation between the error in x_{n+1} and the error in x_n.

7. (A. Ben Israel) The function $f(x) = xe^{-x}$ has a unique zero $\alpha = 0$. Show that for any $x_0 > 1$ the Newton iterates move away from the zero.

8. The Cartesian coordinates of a planet in elliptic orbit at time t are equal to $ea(\sin(x), \cos(x))$, where a is the semi-major axis, and e the eccentricity of the ellipse. Using Kepler's laws of planetary motion it can be shown that the angle x, called the eccentric anomaly, satisfies Kepler's equation

$$x - e\sin x = M, \quad 0 < |e| < 1,$$

where $M = 2\pi t/T$ is the mean anomaly and T the orbital period.

(a) Newton used his method to solve Kepler's equation. Show that for each e, M there is one unique real solution $x = \alpha$, such that $M - |e| \le \alpha < M + |e|$.

(b) Show that the simple fixed-point iteration method

$$x_{n+1} = e\sin x_n + M, \quad x_0 = 0,$$

is convergent.

(c) Study the convergence of Newton's method

$$x_{n+1} = x_n + \frac{e\sin x_n - x_n + M}{1 - e\sin x_n}.$$

8. Determine the multiple root $\alpha = 1$ of the equation $p(x) = (1-x)^5 = 0$, when the function is evaluated using Horner's scheme, i.e.,

$$p(x) = (((((x-5)x + 10)x - 10)x + 5)x - 1 = 0.$$

(a) Use bisection (cf. Algorithm 6.1.1) with initial interval $(0.9, 1.1)$ and tolerance $\tau = 10^{-8}$. What final accuracy is achieved?

(b) Use Newton's method, starting from $x_0 = 1.1$ and evaluating $p'(x)$ using Horner's scheme. Terminate the iterations when for the first time $|x_{n+1} - 1| > |x_n - 1|$. How many iterations are performed before termination? Repeat with a couple of other starting values!

(c) Same as (b), but perform one step of the modified Newton's method (6.3.5) with $x_0 = 1.1$ and $q = 5$. How do you explain that the achieved accuracy is much better than predicted by (6.1.10)?

9. Show that if Newton's method applied to the equation $u(x)=0$, where $u(x)=f(x)/f'(x)$, then

$$x_{n+1}=x_n-\frac{u(x_n)}{1-t(x_n)},\quad t(x_n)=\frac{f(x_n)f''(x_n)}{(f'(x_n))^2}. \tag{6.3.33}$$

This transformation is most useful if an *analytical simplification* can be done such that $u(x)$ can be evaluated accurately also in a neighborhood of α.

10. (a) Show that Halley's method can also be derived by applying Newton's method to the equation $f(x)(f'(x))^{-1/2}=0$.

(b) What is the efficiency index of Newton's and Halley's method, respectively, if it is assumed that evaluating each of f, f' and f'' takes one unit of work.

(c) Show that Halley's method applied to $f(x)=x^2-c=0$, $c>0$, gives rise to the iteration

$$x_{n+1}=x_n\frac{x_n^2+3c}{3x_n^2+c}=x_n-\frac{2x_n(x_n^2-c)}{3x_n^2+c}.$$

Apply Halley's method to $f(x)=x^k-c=0$, $c>0$.

11. (A. Ben Israel) Consider the **quasi-Halley method**

$$x_{n+1}=x_n-\frac{f(x_k)}{f'(x_k)-\dfrac{f'(x_k)-f'(x_{k-1})}{2(x_k-x_{k-1})f'(x_k)}f(x_k)}$$

where the second derivative $f''(x_k)$ has been approximated by a divided difference. Show that if f'' is Lipschitz continuous near a root α then

$$|\alpha-x_{k+1}|=O(|\alpha-x_k|^\gamma,$$

where γ satisfies the quadratic equation $\gamma^2-2\gamma-1=0$. Conclude that the order of this method is approximately 2.41 as compared to 3 for Halley's method.

12. (Bailey et al. [1]) In 1976 Brent and Salamin independently discovered the following iteration, which generates a sequence $\{p_k\}$ converging quadratically to π:
Set $a_0=1$, $b_0=1/\sqrt{2}$, and $s_0=1/2$. For $k=1,2,3,\dots$ compute

$$\begin{aligned}&a_k=(a_{k-1}+b_{k-1})/2,\quad b_k=\sqrt{a_{k-1}b_{k-1}},\\&c_k=a_k^2-b_k^2,\quad s_k=s_{k-1}-2^kc_k,\quad p_k=2a_k^2/s_k.\end{aligned}$$

Perform this iteration in IEEE 754 double precision. Verify the quadratic convergence by listing the errors in $|p_k-\pi|$ in successive iterations. How many iterations can you do before the error starts to grow? What is the best accuracy achieved?

13. In Example 6.3.4 the first two steps in the interval Newton method for solving the equation $x^2-2=0$ are shown. Implement this method and carry out the iterations until convergence.

14. (a) Compute $\det(F_p(x))$ in (6.3.30) for $p = 3$ and write down the corresponding rational fourth order iteration method in terms of u, a_2, a_3 in (6.3.28).

(b) Implement in MATLAB the iteration method (6.3.31) for arbitrary order p. Input should be an approximation x_n to the root, $f(x_n)$ and the row vector of scaled derivatives

$$\left(f'(x),\ \frac{f''(x)}{2!},\ \ldots,\ \frac{f^{(p-1)}(x)}{(p-1)!},\ \frac{f^{(p)}(x)}{p!}\right)$$

evaluated at $x = x_n$. Output should be the diagonal elements of U_p in the LU factorization of $F_p(x_n)$ and the sequence of approximations $x_{n+1,k} = x_n + f(x_n)/u_{kk}(x_n)$, $k = 1 : p$. Try to economize on memory requirement.

15. Write a program that computes the inverse of the error function $\mathrm{erf}(x)$ by solving the equation $\mathrm{erf}(x) - y = 0$, $0 \le y < 1$. Use Newton's method and the series expansion given in Example 1.3.4 to compute values of $\mathrm{erf}(x)$ and its derivative. Note that for large values of x $\mathrm{erf}(x) \approx 1 - 1/(\sqrt{\pi}x)$.

16. (a) Given $\sigma_1 \ge \sigma_1 \ge \ldots \ge \sigma_n > 0$, $c_1, c_2, \ldots c_n$, and $\alpha > 0$, consider the **secular equation** $\phi(\lambda) = \alpha$, where

$$\phi^2(\lambda) = \sum_{i=1}^{n} \left(\frac{\sigma_i c_i}{\sigma_i^2 + \lambda}\right)^2 .$$

Show that $\phi(\lambda)$ is a convex and strictly decreasing function of λ. Conclude that if $\phi(0) > \alpha$, this equation has a unique root $\lambda > 0$ of smallest magnitude.

(b) Newton's method for solving $\phi(\lambda) - \alpha = 0$ is

$$\lambda_{k+1} = \lambda_k - h_k, \quad h_k = \frac{\phi(\lambda_k) - \alpha}{\phi'(\lambda_k)}.$$

Show that with $\lambda_0 = 0$ this method produces a strictly increasing sequence λ_k converging to the solution. Derive an explicit expression for h_k.

(c) The Newton iteration in (b) often converges very slowly. A more efficient method is obtained by instead applying Newton's method to the equation $h(\lambda) = 1/\phi(\lambda) = 1/\alpha$. Show that this iteration can be written

$$\lambda_{k+1} = \lambda_k - h_k \frac{\phi(\lambda_k)}{\alpha},$$

where h_k is the Newton step in (b).

(d) Let $\sigma_i = 1/i^2$, $c_i = 1/i^2 + 0.001$, $i = 1, 2, \ldots, 20$. Plot the function $f(\lambda)$ for $\lambda \in (0, 0.0005)$. Solve the equation $\phi(\lambda) = \alpha = 2.5$, using $\lambda_0 = 0$, comparing the two methods in (b) and (c).

6.4 Local Minimum of a Scalar Function

In this section we consider the problem of finding the minimum (maximum) of a real-valued function

$$\min g(x), \quad x \in [a, b], \tag{6.4.1}$$

which is closely related to that of solving a scalar equation. Problem (6.4.1) occurs as an important subproblem in methods for minimizing a function $\phi(z)$ of n variables, $z \in \mathbf{R}^n$, and for solving systems of nonlinear equation. For example, if z_k is the current approximation to the optimal point, the next approximation is often found by minimizing a function

$$g(\lambda) = \phi(x_k + \lambda d_k),$$

where d_k is a search direction, and the steplength λ is to be determined.

If g is differentiable in $[a, b]$, a **necessary condition** for an interior point of $\tau \in I$ to be a local minimum is that $g'(\tau) = 0$. If g' does not change sign on I it is also possible that the minimum is at a or b. If this is checked for separately, then it is possible to reduce the problem to a zero-finding problem for g'. Since g' also vanishes at a point of maximum and inflection, it is however necessary to check if the point found really is a minimum.

Most algorithms for minimizing a nonlinear function of one (or more) variables find at best a *local* minimum. For a function with several local minima, there is no guarantee that the *global* (lowest local) minimum in $[a, b]$ will be found. One obvious remedy is to try several different starting points and hope that the lowest of the local minima found is also the global minimum. This approach is neither efficient or safe. In practice we have to be content with algorithms which nearly always give correct results in most practical applications.

6.4.1 Unimodal Functions

A condition which ensures that a function g has a unique global minimum τ in $[a, b]$ is that g(x) is strictly decreasing for $a \le x < \tau$ and strictly increasing for $\tau < x \le b$. Such a function is called **unimodal**.

Definition 6.4.1.
The function $g(x)$ is **unimodal** *on $[a, b]$ if there exists a unique $\tau \in [a, b]$ such that, given any $c, d \in [a, b]$ for which $c < d$*

$$d < \tau \Rightarrow g(c) > g(d); \qquad c > \tau \Rightarrow g(c) < g(d). \tag{6.4.2}$$

This condition does not assume that g is differentiable or even continuous on $[a, b]$. For example, $|x|$ is unimodal on $[-1, 1]$.

6.4.2 Golden Section Search

We now describe an **interval reduction method** for finding the local minimum of a unimodal function, which only uses function values of g. It is based on the following lemma.

Lemma 6.4.2. *Suppose that g is unimodal on $[a, b]$, and τ is the point in Definition 6.4.1. Let c and d be points such that $a \le c < d \le b$. If $g(c) \le g(d)$ then $\tau \le d$, and if $g(c) \ge g(d)$ then $\tau \ge c$.*

Proof. If $d < \tau$ then $g(c) > g(d)$. Thus, if $g(c) \le g(d)$ then $\tau \le d$. The other part follows similarly. □

Assume that g is unimodal in $[a, b]$. Then using Lemma 6.4.2 it is possible to find a reduced interval on which g is unimodal by evaluating $g(x)$ at two interior points c and d such that $c < d$. Setting

$$[a', b'] = \begin{cases} [c, b], & \text{if } g(c) > g(d); \\ [a, d], & \text{if } g(c) < g(d). \end{cases}$$

we can enclose x^* in an interval of length at most equal to $\max(b - c, d - a)$. (If $g(c) = g(d)$ then $\tau \in [c, d]$, but we ignore this possibility.) To minimize this length one should take c and d so that $b - c = d - a$. Hence $c + d = a + b$, and we can write

$$c = a + t(b - a), \qquad d = b - t(b - a), \quad 0 < t < 1/2.$$

Then $d - a = b - c = (1 - t)(b - a)$, and by choosing $t \approx 1/2$ we can almost reduce the length of the interval by a factor $1/2$. However, $d - c = (1 - 2t)(b - a)$ must not be too small for the available precision in evaluating $g(x)$.

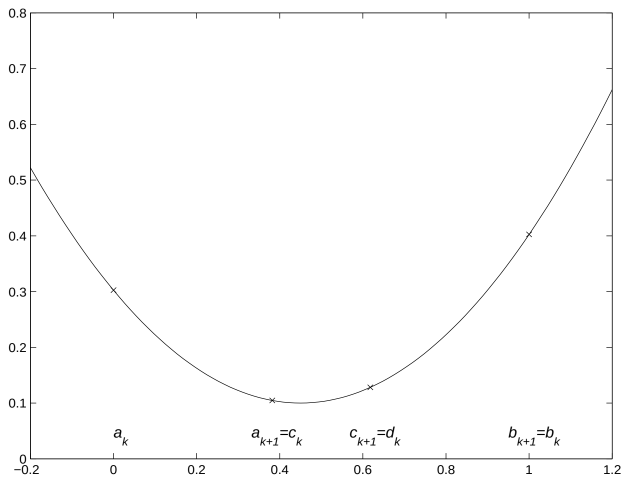

Figure 6.4.1. *One step of interval reduction, $g(c_k) \ge g(d_k)$.*

If we only consider one step the above choice would be optimal. Note that this step requires **two** function evaluations. A clever way to save function evaluations is to arrange it so that if $[c, b]$ is the new interval then d can be used as one of the points in the next step; similarly if $[a, d]$ is the new interval then c can be used at the next step. Suppose this can be achieved with a fixed value of t. Since $c + d = a + b$ the points lie symmetric with respect to the midpoint $\frac{1}{2}(a + b)$ and we need only consider the the first case. Then t must satisfy the following relation (cf. above and Figure 6.7.1)

$$d - c = (1 - 2t)(b - a) = (1 - t)t(b - a).$$

Hence t should equal the root in the interval $(0, 1/2)$ of the quadratic equation $1 - 3t + t^2 = 0$, which is $t = (3 - \sqrt{5})/2$. With this choice the length of the interval will be reduced by the factor

$$1 - t = 2/(\sqrt{5} + 1) = 0.618034\ldots$$

at each step, which is the **golden section** ratio. For example, 20 steps gives a reduction of the interval with a factor $(0.618034\ldots)^{20} \approx 0.661 \cdot 10^{-5}$.

Algorithm 6.4.1 *Golden Section Search.*

Let g be a given continuous function and $I = [a, b]$ an interval. The following algorithm computes an approximation $m \in I$ to a local minimum of $g(x)$, with an error less than a specified tolerance τ.

```
$xmin = goldsec(g, a, b, \tau);$
$t = 2/(3 + \sqrt{5});$
$c = a + t \cdot (b - a);$
$d = b - t \cdot (b - a);$
$gc = g(c); \quad gd = g(d);$
while $(d - c) > \tau \cdot \max(|c|, |d|)$
  if $gc \geq gd$ $\%Keep\ right\ endpoint\ b$
    $a = c; \quad c = d;$
    $d = b - t \cdot (b - a);$
    $gc = gd; \quad gd = g(d);$
  else $\%Keep\ left\ endpoint\ a$
    $b = d; \quad d = c;$
    $c = a + t \cdot (b - a);$
    $gd = gc; \quad gc = g(c);$
  end;
end;
$xmin = (c + d)/2;$
```

Rounding errors will interfere when determining the minimum of a scalar function $g(x)$. Because of rounding errors the computed approximation $fl(g(x))$ of a unimodal function $g(x)$ is not in general unimodal; cf. Figure 6.4.2. However, if we assume that in Definition 6.4.1 the points c and d satisfy $|c - d| > \tau$ for some small τ, the condition (6.4.2) will hold also for the computed function. For any method using only computed values of g there is a fundamental limitation in the accuracy of the computed location of the minimum point τ in $[a, b]$. The best we can hope for is to find $x_k \in [a, b]$ such that

$$g(x_k) \leq g(x^*) + \delta,$$

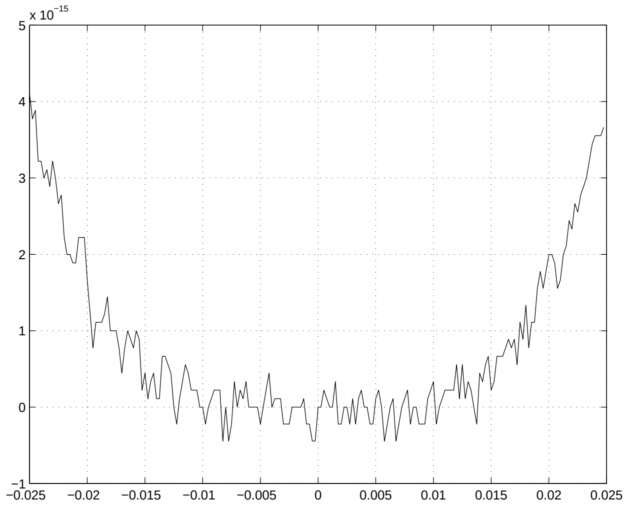

Figure 6.4.2. *The effect of rounding errors on minimizing a function.*

where δ is an upper bound of rounding and other errors in the computed function values $\bar{g}(x) = fl\,(g(x))$; If g is twice differentiable in a neighborhood of a minimum point τ then by Taylor's theorem

$$g(\tau + h) \approx g(\tau) + \tfrac{1}{2}h^2 g''(\tau).$$

This means that there is no difference in the floating point representation of $g(\tau+h)$ unless h is of the order of $\sqrt{u}$. Hence we can not expect τ to be determined with an error less than

$$\epsilon_\alpha = \sqrt{2\,\delta/|g''(x^*)|}, \tag{6.4.3}$$

unless we can also use values of g' or the function has some special form.

6.4.3 Minimization by Interpolation

For finding the minimum of a unimodal function g golden section search method has the advantage that linear convergence is guaranteed. In that respect it corresponds to the bisection method for finding a zero of a function. If the function is sufficiently smooth and we have a good initial approximation, then a process with superlinear convergence will be much faster. Such methods can be devised using interpolation by a polynomial or rational function, chosen so that its minimum is easy to determine. Since these methods do not always converge they should be combined with golden section search. There is a close analogy with robust methods for solving a nonlinear equation, where a combination of inverse interpolation and bisection can be used; see Section 6.2.4.

Since a linear function in general has no minimum the simplest choice is to use a second degree polynomial, i.e. a parabola. Suppose that at step n we have three distinct points in u, v and w. The quadratic polynomial interpolating $g(x)$ at these points is (cf. (6.2.11))

$$p(x) = g(v) + (x - v)[u, v]g + (x - v)(x - u)[u, v, w]g.$$

Setting the derivative of $p(x)$ equal to zero gives

$$0 = [u, v]g + (v - u)[u, v, w]g + 2(x - v)[u, v, w]g.$$

and solving for x

$$x = v + d, \quad d = -\frac{[u, v]g + (v - u)[u, v, w]g}{2[u, v, w]g}. \tag{6.4.4}$$

This is a minimum point of $p(x)$ if $[u, v, w]g > 0$. We assume that of all the points where g has been evaluated v is the one with least function value. Therefore d should be small, so the effect of rounding errors in computing d is minimized. Initially we can take $u = a$, $w = b$, and if $g(c) < g(d)$ then $v = c$ otherwise $v = d$, where c and d are the two golden section points.

Multiplying the nominator and denominator of d by $(v - u)(w - v)(w - u)$, a short calculation shows that $d = -s_1/s_2$, where

$$\begin{aligned} r_1 &= (w - v)(g(v) - g(u)), & r_2 &= (v - u)(g(w) - g(v)), \\ s_1 &= (w - v)r_1 + (v - u)r_2, & s_2 &= 2(r_2 - r_1). \end{aligned} \tag{6.4.5}$$

Consider parabolic interpolation at the points x_{i-2}, x_{i-1}, x_i, $i = 2, 3, \dots$ and let $\epsilon_i = x_i - \tau$. Assuming that $g(x)$ is sufficiently smooth in a neighborhood of τ it can be shown that asymptotically the relation

$$\epsilon_{i+1} \sim \frac{c_3}{2c_2}\epsilon_{i-1}\epsilon_{i-2}, \quad c_r = \frac{1}{k!}g^{(k)}(\zeta_r), \tag{6.4.6}$$

holds between successive errors. Hence the convergence order equals the real root $p = 1.3247\dots$ of the equation $x^3 - x - 1 = 0$.

If two or more of the points u, v, w coincide, or if the parabola degenerates into a straight line, then $s_2 = 0$. The parabolic interpolation step is only taken if the following inequalities are true:

$$|d| < \tfrac{1}{2}|e|, \quad s_2 \neq 0, \quad v + d \in [a, b],$$

where e is the value of the second last cycle and, as before, tol is a combination of absolute and relative tolerance is used

$$\text{tol} = \epsilon|x| + \tau.$$

Otherwise a golden section step is taken, i.e.,

$$x = \begin{cases} (1 - t)v + ta, & \text{if } v \geq \frac{1}{2}(a + b); \\ (1 - t)v + tb, & \text{if } v < \frac{1}{2}(a + b), \end{cases}$$

where $1 - t = 2/(\sqrt{5} + 1)$.

The combination of inverse quadratic interpolation and golden section search has been suggested by Brent [2, 1973, Ch. 5],where the many delicate points to consider in an implementation are discussed. At a typical step there are six significant points a, b, u, v, w and x, not all distinct. The position of these points are updated at each step. Initially $[a, b]$ is an interval known to contain a local minimum point. At a later point in the algorithm they have the following significance: A local minimum lies in $[a, b]$; of all the points at which g has been evaluated v is the one with the least value of g; w is the point with the next lowest value of g; u is the previous value of w, and x is the last point at which g has been evaluated.

The Matlab function `fminbnd` is based on the Fortran implementation FMIN of Brent's algorithm given in Forsythe, Malcolm, and Moler [5, pp.184–187].

Review Questions

1. How many steps in needed in golden section search to reduce an initial interval $[a, b]$ by a factor of 10^{-6}?
2. Suppose the twice differentiable function $f(x)$ has a local minimum at a point x^*. What approximate limiting accuracy can you expect in a method for computing x^* which uses only function values?
3. The algorithm FMIN is a standard method for finding the minimum of a function. It uses a combination of two methods. Which?

Problems and Computer Exercises

1. Use the algorithm goldsec to find the minimum of the quadratic function $f(x) = (x - 1/2)^2$ starting from $a = 0.25$, $b = 1$. Plot the successive inclusion intervals.
2. Modify the algorithm goldsec to use parabolic interpolation instead of golden section if this gives a point within the interval.
3. (a) Plot the function

$$g(x) = \frac{1}{(x - 0.3)^2 + 0.01} + \frac{1}{(x - 0.9)^2 + 0.04},$$

and show that it has a local minimum in each of the intervals $[0.2, 0.4]$ and $[0.8, 1.0]$.

(b) Use your algorithm from Problem 2 to determine the location of the two minima of $g(x)$ in (a).

(c) MATLAB includes a function `fminbnd`, that also uses a combination of golden section search and parabolic interpolation to find a local minimum. Compare the result using this function with the result from (b).

4. (Brent [2, Sec. 5.6]) The function

$$g(x) = \sum_{i=1}^{20} \left(\frac{2i-5}{x-i^2} \right)^2,$$

has poles at $x = 1^2, 2^2, \dots, 20^2$. Restricted to the open interval $(i^2, (i+1)^2)$, $i = 1 : 19$, it is unimodal. Determine the minimum points in these intervals and the corresponding values of $g(x)$.

6.5 Zeros of Polynomials

6.5.1 Introduction

The problem of solving a polynomial equation

$$p(z) = a_0 z^n + a_1 z^{n-1} + \cdots + a_n = 0, \quad (a_0 \neq 0), \tag{6.5.1}$$

has played a major role in the development of mathematical concepts for many centuries. Even the "high school formula" for solving a quadratic equations requires the introduction of irrational and complex numbers. There is a long history of investigations into algebraic expressions for the zeros of equations of higher degree. In the 16th century Cardano published formulas for the roots of a cubic equation (see Problem 2.3.8). Formulas for the roots when $n = 4$ are also known. In 1826 Abel proved that it is not possible to find algebraic expressions for the roots for the class of polynomial equations of degree $n > 4$. However, even the existing formulas for $n \leq 4$ are not in general suitable for numerical evaluation of the roots. In Section 2.3.2, it was shown that care must be taken to ensure numerical stability also for the simple formulas in the quadratic case (see also Problem 2)!

Despite the absence of closed solution formulas the **fundamental theorem of algebra** states for any algebraic equation $p(z) = 0$ of degree $n > 0$, there exists at least one complex number z_1 such that $p(z_1) = 0$. Hence we can write $p(z) = p_0(z) = (z - z_1)p_1(z)$, where $p_1(z)$ is a polynomial of degree $n-1$. Now $p_1(z)$ must have at least one root z_2, and we write $p_1(z) = (z - z_2)p_2(z)$, where $p_2(z)$ has degree $n - 2$. Continuing this reasoning $p_{n-1}(z) = (z - z_n)p_n$, where $p_n = a_n$ is a constant. It follows that, counting multiplicities, the equation (6.5.1) has exactly n (real or complex) roots $z_1, z_2, \dots, z_n$, and

$$p(z) = a_0(z - z_1)(z - z_2)\cdots(z - z_n). \tag{6.5.2}$$

By this representation it also follows that if the coefficients $a_0, a_1, \dots, a_n$ are real, then eventual complex roots must occur in conjugate pairs.

Solving polynomial equations of high degree does not play a central role in scientific computing. Usually the applications involve only equations of moderate degree, say 10–20, for which acceptable subroutines exist. Polynomial equations of high degree play a major role also as a computational task in the area of computer algebra, where one need to solve (6.5.1) for $n > 100$. This in general requires high multiple precision computations and algorithm for such problems are still a subject of research.

It should be emphasized that although a problem may be given in the form (6.5.1), it could be that the coefficients of $p(z)$ are not the original data. Then it may be better to avoid computing them. An important example of this is when the polynomial is the **characteristic polynomial**

$$p_{(}z) = \det(zI - A)$$

of a matrix $A \in \mathbf{A}^{n\times n}$. Then it is an eigenvalue problem in disguise, and the n roots of $p_A(z) = 0$ are the eigenvalues of A. Here the original data are the elements of the matrix A and numerical values of $p(z)$ can then in general be evaluated much more accurately directly from the matrix elements, see Chapter 9. Even when the eigenvalues are well determined by the elements of A and appear to be well separated, they can be *extraordinary sensitive to to small relative perturbations in the coefficients coefficients of* $p_A(z)$. In the following we discuss a famous example, due to Wilkinson [28, 1984]. This paper[7] contains an extensive discussion of numerical problems in determining roots of polynomial equations.

Example 6.5.1.

Consider the Wilkinson polynomial

$$p(z) = (z-1)(z-2)\cdots(z-20) = z^{20} - 210z^{19} + \ldots + 20!,$$

with zeros $1, 2, \ldots, 20$. Let $\bar{p}(z)$ be the polynomial which is obtained when the coefficient $a_1 = 210$ in $p(z)$ is replaced by

$$-(210 + 2^{-23}) = -210.000000119\ldots,$$

while the rest of the coefficients remain unchanged. Even though the relative perturbation in a_1 is of order 10^{-10}, many of the zeros of the perturbed polynomial $\bar{p}(z)$ deviate greatly from those of $p(z)$. In fact, correct to nine decimal places, the perturbed zeroes are

1.000000000	$10.095266145 \pm 0.643500904i$
2.000000000	
3.000000000	$11.793633881 \pm 1.652329728i$
4.000000000	
4.999999928	$13.992358137 \pm 2.518830070i$
6.000006944	
6.999697234	$16.730737466 \pm 2.812624894i$
8.007267603	
8.917250249	$19.502439400 \pm 1.940330347i$
20.846908101	

For example, the two zeros $16, 17$ have not only changed substantially, but have become a complex pair. It should be emphasized that this behavior is quite typical

[7]Wilkinson received the Chauvenet Prize of the Mathematical Association of America for this exposition of the ill-conditioning of polynomial zeros.

of polynomials with real coefficients and real roots. Indeed, many polynomials which arise in practice behave much worse than this.

If we assume that the coefficients a_i of a polynomial are given with full machine accuracy, then the error δ in computed values of $p(x)$ (for real x) is bounded by

$$\delta < 1.06u \sum_{i=0}^{n} |(2i+1)a_{n-i}x^i| < \gamma_{2n+1} \sum_{i=0}^{n} |a_{n-i}||x|^i,$$

see Section 2.4. Hence by (6.1.7) the attainable accuracy of a zero α is equal to

$$\epsilon_\alpha = \frac{\delta}{|p'(\alpha)|} = \frac{\sum_{i=0}^{n} |(2i+1)a_{n-i}\alpha^i|}{|p'(\alpha)|}.$$

In particular for the root $\alpha = 14$ in the above example we get $\epsilon_\alpha = 1.89 \cdot 10^{16}$. However, the changes in this example are so large that this linearized perturbation theory does not apply!

As the above example emphasizes, *computing the characteristic polynomial is not, as is sometimes thought, a simplification of the eigenvalue problem.* Eigenvalue problems should be solved with one of the highly developed modern eigenvalue algorithms; consult Chapter 9, Volume II, and references therein! Note also that if the coefficients of the characteristic polynomial $\det(A - zI)$ are required, these are best computed by first computing the eigenvalues λ_i of A are computed and then forming

$$p(z) = \prod_{i=1}^{n} (z - \lambda_i). \tag{6.5.3}$$

The **companion matrix** of the polynomial $p(z)$ in (6.5.1), normalized so that $a_0 = 1$, is defined as

$$C = \begin{pmatrix} -a_1 & -a_2 & \cdots & -a_{n-1} & -a_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}. \tag{6.5.4}$$

(Sometimes the companion matrix is defined slightly differently, e.g., with the coefficients of the polynomial in the last row or in the last column.) Using the definition (1.6.4) it can be verified that the characteristic polynomial of C equals

$$p_C(z) = \det(zI - C) = z^n + a_1 z^{n-1} + \cdots + a_{n-1}z + a_n.$$

Hence the the eigenvalues of C are the roots of $p(z) = 0$ and turning the tables, algorithms for solving eigenvalue problems can be used for solving polynomial equations. In MATLAB the function `roots(p)` computes the roots of a polynomial $p(z)$ using the QR algorithm to solve the eigenvalue problem for the companion matrix. Although the operation count for this QR algorithm is $\mathcal{O}(n^3)$ and the storage requirement $1.5n^2$ experiments suggest that for small and moderate values of n it is as as fast as competing algorithms and can be more accurate. Further problems with overflow or underflow are avoided. However, it seems less suitable for applications where n is large and the roots are wanted to multiple precision accuracy.

If the coefficients are known (and stored) exactly, then by using multiple-precision arithmetic the accuracy in the zeros can be increased. It is generally true that the solution of polynomial equations of high degree requires the use of multiple precision floating-point arithmetic in order to achieve high accuracy.

Example 6.5.2.

The largest positive root of the equation

$$p(x) = (x + 2)(x^2 - 1)^6 - 3 \cdot 10^{-6} \cdot x^{11} = 0$$

is to be computed. Here $p(z)$ is a polynomial of degree 13. If the coefficients are computed using decimal floating point arithmetic with seven digits, then the coefficient of x^{11} which is $(12 - 3 \cdot 10^{-6})$ will be rounded to 12.00000. Thus the machine will treat the equation $(x+2)(x^2-1)^6 = 0$, whose exact positive root is 1.

This is a poor result. One can get the root $\alpha = 1.053416973823$ to full accuracy for example by writing the equation in the form

$$x = \phi(x), \qquad \phi(x) = 1 + \frac{0.1}{x+1}\left(\frac{3x^{11}}{x+2}\right)^{1/6},$$

and solving this by the iteration $x_0 = 1$, $x_{k+1} = \phi(x_k)$. Hence the relative error in the previous result is greater then 5%.

6.5.2 Some Basic Formulas

Comparing the coefficients of z^{n-k} in the representations (6.5.1) and (6.5.2) of $p(z)$ we find that $(-1)^k a_k/a_0$ is the sum of the $\binom{n}{k}$ products of the roots taken k at a time. Thus we obtain the following relations between the coefficients and zeros of a polynomial

$$\begin{aligned} &\sum_i z_i = -a_1/a_0, \quad \sum_{i<j} z_i z_j = \ a_2/a_0, \quad \sum_{i<j<k} z_i z_j z_k = \ a_3/a_0, \cdots \\ &\cdots z_1 z_2 \cdots z_n = (-1)^n a_n/a_0. \end{aligned} \tag{6.5.5}$$

The functions on the left side are called **elementary symmetric functions** of the variables $z_1, z_2, \ldots, z_n$, since interchanging any of the variable will not change the functions.

If $a_n \neq 0$ then the **reciprocal polynomial** is

$$q(y) = y^n p(1/y) = a_n y^n + \cdots + a_1 y + a_0. \tag{6.5.6}$$

The zeros of the reciprocal polynomial are $1/z_1, 1/z_2, \ldots, 1/z_n$ and from (6.5.5) we have the relations,

$$\sum_i 1/z_i = -a_{n-1}/a_n, \quad \text{etc..}$$

Function values of the polynomial $p(z)$ in (6.5.1) at a (real or complex) point w can conveniently be computed by repeated **synthetic division** of $p(z)$ with $z-w$; cf. Section 1.3.1. Let

$$\begin{aligned} p(z) &= (z-w)q(z)+b_n, \\ q(z) &= b_0z^{n-1}+b_1z^{n-2}+\ldots+b_{n-1}. \end{aligned} \tag{6.5.7}$$

Then the sequence $\{b_i\}_{i=0}^n$ is defined by the recursion

$$b_0=a_0, \quad b_i=b_{i-1}w+a_i, \quad i=1:n. \tag{6.5.8}$$

Setting $z=w$ we we see that $p(w)=b_n$ is the remainder and $q(z)$ is the quotient polynomial when dividing $p(z)$ with $(z-w)$. Differentiating (6.5.7) we get

$$p'(z)=(z-w)q'(z)+q(z), \tag{6.5.9}$$

and setting $z=w$ we find that $p'(w)=q(w)$. We can evaluate $q(w)$ by synthetic division of $q(z)$ with $(z-w)$,

$$\begin{aligned} q(z) &= (z-w)r(z)+c_{n-1}, \\ r(z) &= c_0z^{n-2}+c_1z^{n-3}+\ldots+c_{n-2}. \end{aligned}$$

Now $p'(w)=q(w)=c_{n-1}$, where

$$c_0=b_0, \qquad c_i=c_{i-1}w+b_i, \quad i=1:n-1.$$

Higher derivatives can be computed in the same fashion. Differentiating once more gives

$$p''(z)=(z-w)q''(z)+2q'(z),$$

and so $\frac{1}{2}p''(w)=q'(w)=d_{n-2}$, where

$$d_0=c_0, \qquad d_i=d_{i-1}w+c_i, \quad i=1:n-2.$$

To compute $p^{(i)}(w)$ using these formulas requires $n-i$ additions and multiplications.

In the important special case where all the coefficients $a_0,a_1,\ldots,a_n$ are real, the above formulas are somewhat inefficient, and one can save operations by performing synthetic division with the quadratic factor

$$(z-w)(z-\bar{w})=z^2-2zRe(w)+|w|^2,$$

which has real coefficients (see Problem 1).

Synthetic division can also be used to shift the origin of a polynomial $p(z)$. Given $a_0,a_1,\ldots,a_n$ and s, we then want to find coefficients $c_0,c_1,\ldots,c_n$ so that

$$p(w+s)=q(s)=c_0s^n+c_1s^{n-1}+\cdots+c_n. \tag{6.5.10}$$

Clearly this is the Taylor expansion of $p(z)$ at $z=w$. It follows that

$$c_n=p(w), \quad c_{n-1}=p'(w), \quad c_{n-1}=\frac{1}{2}p''(w), \quad \ldots, \quad c_0=\frac{1}{n!}p^{(n)}(w),$$

and the coefficients c_i can be computed by repeated synthetic division of $p(z)$ by $(z-w)$ as described above in about $n^2/2$ multiplications.

It is often desirable to obtain some preliminary information as to where the zeros of a polynomial $p(z)$ are located. Some information about the location of real roots can be obtained from a simple examination of the sign of the coefficients of the polynomial. A simple observation is that if $a_i > 0$, $i = 1 : n$, then $p(x)$ can have no real positive zero. A generalization of this result, known as **Descartes' rule of sign**, [8] states that the number of positive roots is either given by the number of variations in sign in the sequence $a_0, a_1, \dots, a_n$, or is *less* than that by an even number. (Multiple roots are counted with their multiplicity.) By considering the sign variations for the polynomial $p(-z)$ we similarly get an upper bound on the number of negative roots. By shifting the origin, we can get bounds on the number of roots larger and smaller than a given number. In Sec. 6.5.3 we give a method to obtain precise information about the number of real roots in any given interval.

Many classical results are known about the number of real or complex roots in a disk or half plane. It is outside the scope of this presentation and we refer to surveys in the literature.

6.5.3 Sturm Sequences

Precise information about the number of real roots in an interval can be obtained from a **Sturm sequence**[9] for $p(x)$.

Definition 6.5.1.

A sequence of real polynomials $p_0(x), p_1(x), \dots, p_m(x)$ is a strict Sturm sequence for $p(x) = p_0(x)$ on the interval $[a, b]$ if the following conditions hold:

(i) *No two consecutive polynomials in the sequence vanish simultaneously on the interval $[a, b]$.*

(ii) *If $p_j(r) = 0$ for $j < m$, then $p_{j-1}(r)p_{j+1}(r) < 0$.*

(iii) *Throughout the interval $[a, b]$, $p_m(x) \neq 0$.*

(iv) *If $p_0(r) = 0$, then $p_0'(r)p_1(r) > 0$.*

Given a polynomial $p_1(x)$ of degree not greater than that of $p_0(x)$ a Sturm sequence can be constructed by the Euclidean algorithm as follows. Let $q_1(x)$ be the quotient polynomial and $-p_2(x)$ the remainder in the quotient $p_0(x)/p_1(x)$, i.e. $p_0(x) = q_1(x)p_1(x) - p_2(x)$, where the degree of $p_2(x)$ is strictly less than that of $p_1(x)$. Continuing in this way, we compute $p_2(x), \dots, p_m(x)$ by

$$p_{k+1}(x) = q_k(x)p_k(x) - p_{k-1}(x), \quad k = 1 : m-1, \qquad (6.5.11)$$

[8] René Descartes (1596–1650) French philosopher and mathematician.

[9] J. C. F. Sturm (1803–1855) a Swiss mathematician best known for his theorem on Sturm sequences, discovered in 1829 and his theory of Sturm–Liouville differential equations. He succeeded Poisson in the chair of mechanics at the École Polytechnique in Paris 1839.

where $q_k(x)$ is the quotient and $-p_{k+1}$ the remainder in the quotient $p_{k-1}(x)/p_k(x)$. We stop when $p_m(x)$ nowhere vanishes on the interval $[a, b]$. Clearly, if $p_i(r) = 0$, then $p_{j+1}(r) = -p_{j-1}(r) < 0$, so condition (ii) is satisfied.

Let $V(x)$ denote the number of variations in sign in the Sturm sequence at x. If $p_0(x)$ and $p_1(x)$ have only simple zeros that separate each other, then it can be shown that the number of zeros of $p_0(x)$ on $[a, b]$ is equal to $|V(a) - V(b)|$.

Theorem 6.5.2.

Take $p_1(x) = p_0'(x)$ and define $p_2(x), \ldots, p_m(x)$ by (6.5.11), where $p_m(x)$ has a fixed sign on the interval $[a, b]$ ($p_0(a) \neq 0$ and $p_0(b) \neq 0$). Let V(r) denote the number of variations of sign in the sequence of values $p_0(r), p_1(r), \ldots, p_m(r)$, vanishing terms not being counted. Then the number of roots of $p_0(x)$ in $[a, b]$, each multiple root being counted once, is exactly equal to $|V(a) - V(b)|$.

Note that if all real zeros of $p_0(x)$ are simple and $p_1(x) = p_0'(x)$, then (6.5.11) generates a Sturm sequence. If $p_0(x)$ has multiple zeros, then $p_0(x)$ and $p_0'(x)$ have a common divisor, which divides every $p_i(x)$ in the sequence, and this will not affect $V(r)$.

Example 6.5.3.

The equation $p(x) = p_0 = x^5 - 3x - 1 = 0$ has three real roots $z_1 = -1.21465$, $z_2 = -0.33473$, and $z_3 = 1.38879$ and two complex roots. The derivative equals $p'(x) = p_1 = 5x^4 - 3$, and the rest of the Sturm chain is given by

$$p_2 = \frac{12}{5}x + 1, \quad p_3 = \frac{59083}{20736}.$$

Here p_2 is a polynomial of degree one and the Sturm chain ends with $s = 3 < n$.

We denote by $[l_k, u_k]$ an interval containing the zero x_k. Evaluating the sign changes of the Sturm sequence at $x = -2$ and $x = 2$ shows that there are $3 - 0 = 3$ roots x_k, $k = 1, 2, 3$, in the interval $[-2, 2]$ Counting the number of sign changes at the midpoint $x = 0$ allows us to deduce that $u_k = 0$, $k = 1, 2$ and $l_3 = 0$; see Table 6.5.1. The interval $[-2, 0]$ contains two roots so we determine next the number of sign changes at the midpoint $x = -1$.

At this point we have determined three disjoint intervals $[-2, -1]$, $[-1, 0]$, and $[0, 2]$, which each contain one root. We continue bisecting each of these intervals, which can be performed in parallel.

Methods based on Sturm sequences can can be competitive, when only a relatively small number of real roots in an given interval are of interest. Consider a real symmetric tridiagonal matrix,

$$A = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{n-1} & \alpha_{n-1} & \beta_n \\ & & & \beta_n & \alpha_n \end{pmatrix},$$

Table 6.5.1. *Left: Sign variations in the Sturm sequence. Right: Intervals* $[l_k, u_k]$ *containing the zero* x_k*.*

x	p_0	p_1	p_2	p_3	δ
-2	$-$	$+$	$-$	$+$	3
$+2$	$+$	$+$	$+$	$+$	0
0	$-$	$-$	$+$	$+$	1
-1	$+$	$+$	$-$	$+$	2
1	$-$	$+$	$+$	$+$	1

l_1	u_1	l_2	u_2	l_3	u_3
-2	2	-2	2	-2	2
	0		0	0	
	-1	-1			
				1	

such that $\beta_k \neq 0$, $k = 2 : n$ has only simple eigenvalues. Let $p_k(\lambda)$ be the characteristic polynomial of the kth leading principal minor of $(A - \lambda I)$. Define $p_0(\lambda) = 1$, and $p_k(\lambda)$ by the three-term recursion

$$p_1(\lambda) = \alpha_1 - \lambda, \quad p_k(\lambda) = (\alpha_k - \lambda)p_{k-1}(\lambda) - \beta_k^2 p_{k-2}(\lambda), \quad k = 2 : n. \tag{6.5.12}$$

Then the sequence

$$1, p_0(\lambda), \dots, p_n(\lambda) = \det(A - \lambda I)$$

is known to form a Sturm sequence. Combined with the bisection method, this recursion can be used to develop an efficient numerical method for determining the eigenvalues of a symmetric tridiagonal matrix A in a given interval $[a, b]$ without reference to any of the others; see Vol. II, Sec. 9.6.6. It can also be used for determining the singular values of a bidiagonal matrix in a given interval; see Vol. II, Sec. 9.7.7.

The Sturm sequence algorithm only works when $f(x)$ is a real function of a real variable. To determine complex zeros an algorithm that performs a search and exclusion of the complex plane can be used. The **quadtree** exclusion algorithm, due to H. Weyl [27], and illustrated in Figure 6.5.1, is such a "two-dimensional bisection algorithm".[10] It was one of the first algorithms with guaranteed convergence to all n zeros of a polynomial of degree n. The algorithm is based on a exclusion test applied to squares in the complex plane. For example, assuming that $f(z)$ is analytic in K then if $|f'(z)| \leq M$ for all $z \in K$ and $|f(z_0)| > \eta M$ there is no zero of $f(z)$ in K. Any square that does not pass the test and thus may contain a root is called suspect. (Note that it is not required that a suspect square actually contains a root.)

The computations begin with an initial suspect square S containing all the zeros of $p(x)$. This square can be found from an upper bound on the absolute value of the zeros of $p(x)$. In the algorithm, as soon as we have a suspect square, this is partitioned into four congruent subsquares. At the center of each of them a test estimating the distance to the closest zero of $p(x)$ is performed. (A relative error within, say, 40% will suffice.) If the test guarantees that this distance exceeds half of the length of the diagonal of the square then the square cannot contain any

[10] In general a quadtree is a tree where each node is split along d dimensions giving 2^d children.

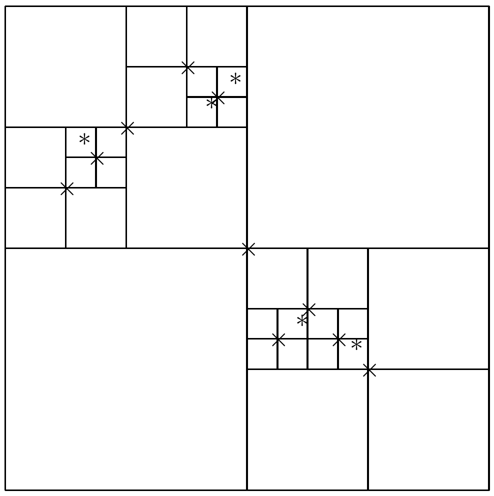

Figure 6.5.1. *Suspect squares computed by Weyl's quadtree method. Their centers (marked by $\times$) approximate the five zeros marked by $*$.*

zero and is discarded. Each remaining suspect square undergoes the same recursive partitioning into four subsquares and the test. The zeros lying in a suspect square are approximated by its center with errors bounded by half the length of its diagonal. Each iteration step decreases the diagonal of the remaining squares by a factor of two so the errors will decrease by a factor of 1/2.

6.5.4 Deflation and Zero Suppression

Suppose we have found a root α to the equation $p(z) = 0$. Then taking $z_k = \alpha$ in (6.5.8)–(6.5.7) we have $b_n = p(\alpha) = 0$ and the remaining roots of $p(z)$ are also roots of the polynomial equation

$$q(z) = \frac{p(z)}{z - \alpha} = 0$$

Hence we can continue the iterations with the quotient polynomial $q(z)$ of degree $n - 1$. This process is called **deflation** and can be repeated; as soon as a root has been found it is factored out. Proceeding like this, all roots are eventually found. Since we work with polynomials of lower and lower degree, deflation saves arithmetic operations. More important is that it prevents the iterations to converge to the same simple root more than once.

So far we have ignored that roots which are factored out are only known with finite accuracy. Also rounding errors occur in the computation of the coefficients of the quotient polynomial $q(x)$. Clearly there is a risk that both these types of errors can have the effect that the zeros of the successive quotient polynomials deviate more and more from those of $p(z)$. Indeed, deflation is not unconditionally a stable numerical process. A closer analysis performed by Wilkinson [28, 1963][11] shows

[11]James Hardy Wilkinson, English mathematician 1919–1986. From 1946 Wilkinson he worked

that if the coefficients of the quotient polynomials are computed by the recursion (6.5.8), then errors resulting from deflation are negligible provided that:

1. the roots are determined in order of *increasing* magnitude;
2. each root is determined to its limiting accuracy.

Note that if the above procedure is applied to the reciprocal polynomial $z^n p(1/z)$ we obtain the zeros of $p(z)$ in order of *decreasing* magnitude.

With Laguerre's method it is quite probable that we get convergence to the root of smallest magnitude from the initial value $z_0 = 0$. However, this cannot be guaranteed and therefore one often proceeds in two steps. First, all n roots are determined using deflation in the process. Next, each root found in the first step is refined by doing one or several Newton iterations using the *original* polynomial $p(z)$.

Deflation can be avoided by using a **zero suppression** technique suggested by Maehly [1954]. He notes that the derivative of the reduced polynomial $q(z) = p(z)/(z - \xi_1)$ can be expressed as

$$q'(z) = \frac{p'(z)}{z - \xi_1} - \frac{p(z)}{(z - \xi_1)^2}.$$

More generally, assume that we have determined approximations $\xi_1, \dots, \xi_j$ to j roots of $p(z) = 0$. Then the the first derivative of the reduced polynomial $q_j(z) = p(z)/[(z - \xi_1) \cdots (z - \xi_j)]$ can be expressed as

$$q_j'(z) = \frac{p'(z)}{(z - \xi_1) \cdots (z - \xi_j)} - \frac{p(z)}{(z - \xi_1) \cdots (z - \xi_j)} \sum_{i=1}^{j} \frac{1}{z - \xi_i}.$$

Hence Newton's method applied to $q_j(z)$ can be written

$$z_{k+1} = z_k - \frac{p(z_k)}{p'(z_k) - \sum_{i=1}^{j} p(z_k)/(z_k - \xi_i)}, \tag{6.5.13}$$

which is the **Newton–Maehly method**. This iteration has the advantage that it is not sensitive to the accuracy in the approximations to the previous roots $\xi_1, \dots, \xi_j$. Indeed, the iteration (6.5.13) is locally quadratically convergent to simple zeros of $p(z)$ for *arbitrary* values of $\xi_1, \dots, \xi_j$.

6.5.5 Simultaneous Determination of Roots

For the removal of a linear factor by deflation it is necessary that the zero has been computed to full working accuracy, since otherwise the remaining approximative zeros can be meaningless. This is a disadvantage if only low accuracy is required.

on the group that built the Pilot ACE computer at National Physical Laboratory, first as Turing's assistant and later as manager of the group. He later contributed greatly to the development of reliable software for matrix computations.

An alternative to deflation is to use an iterative methods that, under appropriate separation assumptions, allows for the *simultaneous* determination of all the roots of a polynomial equation. Suppose that the numbers $\xi_i^{(k)}$, $i = 1 : n$ are a set of n distinct approximations to the of $p(z)$. A new set of approximations are then computed from

$$\xi_i^{(k+1)} = \xi_i^{(k)} - p(\xi_i^{(k)}) \Big/ \Big[a_0 \prod_{\substack{j=1 \\ j\neq i}}^{n} (\xi_i^{(k)} - \xi_j^{(k)}) \Big], \quad i = 1 : n. \tag{6.5.14}$$

This is **Weierstrass' method**,[12] introduced in 1891 in connection with a new constructive proof of the fundamental theorem of algebra. The method was rediscovered and analyzed in the 1960s by Durand and is also known as the **Durand–Kerner method**.

With $q(z) = (z - \xi_1^{(k)})(z - \xi_2^{(k)}) \cdots (z - \xi_n^{(k)})$ the formula may also be written

$$\xi_i^{(k+1)} = \xi_i^{(k)} - p(\xi_i^{(k)})/q'(\xi_i^{(k)}),$$

which shows that to first approximation the method is identical to Newton's method. This relation can be used to prove that for simple (real or complex) zeros the asymptotic order of convergence of the Weierstrass method equals 2. (For multiple zeros the method will only converge linearly.) The relation

$$\sum_{i=1}^{n} \xi_i^{(k)} = \sum_{i=1}^{n} \alpha_i = -a_1, \quad k \geq 1,$$

which holds independent of the initial approximations, can be used as a control; see Kjellberg [12].

It is possible to accelerate Weierstrass method by using the new approximations of the roots in (6.5.14) as they become available. This leads to the iteration

$$\xi_i^{(k+1)} = \xi_i^{(k)} - p(\xi_i^{(k)}) \Big/ \Big[a_0 \prod_{j<i} (\xi_i^{(k)} - \xi_j^{(k+1)}) \prod_{j>i} (\xi_i^{(k)} - \xi_j^{(k)}) \Big], \quad i = 1 : n.$$

This *serial* version of the Weierstrass method can be shown to have an order of convergence at least $1 + \sigma_n$, where $1 < \sigma_n < 2$ is the unique positive root to $\sigma^n - \sigma - 1 = 0$.

If no a priori information about the roots is available then the initial approximations $\xi_i^{(0)}$ can be chosen equidistantly on a circle $|z| = \rho$, centered at the origin, which encloses all the zeros of $p(z)$. Such a circle can be found, e.g., by using the result that all the roots of the polynomial $p(z)$ lie in the disk $|z| \leq \rho$, where

$$\rho = \max_{1\leq k\leq n} 2 \left(\frac{|a_k|}{|a_0|} \right)^{1/k}.$$

Note that this is (6.5.15) applied to the reciprocal polynomial.

[12]Karl Theodor Wilhelm Weierstrass (1815–1897) influential German mathematician, often said to be the father of modern analysis.

6.5.6 A Modified Newton Method

Most of the iteration methods described in earlier sections can be applied to polynomial root finding. Note that if $p(z)$ has real coefficients, then $p(z)$ and $p'(z)$ are real for real values of z. This means that Newton's method cannot converge to a *complex* root from a *real* initial approximation. The same holds for the secant method and its variants. The Muller–Traub method (see Section 6.2.3) can also converge to complex roots from real approximations. It requires only one evaluation of $p(z)$ per step, and has therefore become popular.

If we have sufficiently good initial approximations to a (real or complex) zero of $p(z)$, this can be computed by Newton's method. It is desirable to find the zeros in roughly increasing order of magnitude, since this leads to stable deflation; see Sec. 6.5.4. We now describe a modified Newton method due to Madsen [14], which has been shown to be very competitive. By including a one-dimensional search along the Newton direction this method achieves good global convergence properties and is effective also for multiple roots.

To initialize let $z_0 = 0$,

$$\delta z_0 = \begin{cases} -p(0)/p'(0) = -a_n/a_{n-1}, & \text{if } a_{n-1} \neq 0 \\ 1 & \text{otherwise,} \end{cases}$$

and take

$$z_1 = \frac{1}{2\rho}\frac{\delta z_0}{|\delta z_0|}, \qquad \rho = \max_{1\le k\le n}\left(\frac{|a_{n-k}|}{|a_n|}\right)^{1/k}. \tag{6.5.15}$$

This assures that $|z_1|$ is less than the modulus of any zero of $p(z)$ (see [10, Exercise 2.2.11]). Further, if $p'(0) \neq 0$, it is in the direction of steepest descent of $|p(z)|$ from the origin (see Sec. 6.3.2). This choice makes it likely that convergence will take place to a root of near minimal modulus.

The general idea of the algorithm is that given z_k, a tentative step h_k is computed by Newton's method. The next iterate is found by taking the best point (in terms of minimizing $|f(z)|$) found by a short search along the line through z_k and $z_k + h_k$. When the search yields no better value than at z_k we take $z_{k+1} = z_k$ and make sure that the next search is shorter and in a different direction. Since the line searches will be wasteful if we are near a simple root, we then switch to the standard Newton's method.

In the first stage of the algorithm, when searches are being performed, new iterates z_{k+1} are computed as follows:

1. If the last iteration was successful ($z_k \neq z_{k-1}$) then the Newton correction
$$h_k = -p(z_k)/p'(z_k), \tag{6.5.16}$$
is computed and the next tentative step is taken as
$$\delta z_k = \begin{cases} h_k, & \text{if } |h_k| \le 3|z_k - z_{k-1}|; \\ 3|z_k - z_{k-1}|e^{i\theta}h_k/|h_k| & \text{otherwise.} \end{cases}$$
Here θ is chosen rather arbitrarily as $\arctan(3/4)$. This change of direction is included because if a saddle point is being approached, the direction h_k may be a bad choice.

2. If the last step was unsuccessful ($z_k = z_{k-1}$) we change the search direction and reduce the step size. In this case the tentative step is chosen to be

   $$\delta z_k = -\tfrac{1}{2} e^{i\theta} \delta z_{k-1}.$$

   Repeated use of this is sure to yield a good search direction.

3. Once the tentative step δz_k has been found we test the inequality $|p(z_k + \delta z_k)| < |p(z_k)|$. If this is satisfied we calculate the numbers

   $$|p(z_k + p\,\delta z_k)|, \quad p = 1, 2, \ldots, n,$$

   as long as these are strictly decreasing. Note that, if we are close to a multiple root of multiplicity m, then we will find the estimate $z_k + mh_k$, which gives quadratic convergence to this root. A similar situation will hold if we are at a fair distance from a cluster of m zeros and other zeros are further away.

   If $|p(z_k + \delta z_k)| \geq |p(z_k)|$, we calculate the numbers

   $$|p(z_k + 2^{-p} \delta z_k)|, \quad p = 0, 1, 2,$$

   again continuing until the sequence ceases to decrease.

A switch to standard Newton is made if in the previous iteration a standard Newton step $z_{k+1} = z_k + h_k$ was taken and Theorem 6.3.2 ensures the convergence of Newton's method with initial value z_{k+1}, i.e., when $f(z_k)f'(z_k) \neq 0$ and

$$2\,|f(z_k)| \max_{z \in K_k} |f''(z)| \leq |f'(z_k)|^2, \quad K_k : |z - z_k| \leq |h_k|,$$

is satisfied, cf. (6.3.12). This inequality can be approximated using already computed quantities by

$$2\,|f(z_k)||f'(z_k) - f'(z_{k-1})| \leq |f'(z_k)|^2 |z_{k-1} - z_k|. \tag{6.5.17}$$

The iterations are terminated and z_{k+1} accepted as a root whenever $z_{k+1} \neq z_k$ and

$$|z_{k+1} - z_k| < u|z_k|,$$

holds, where u is the unit roundoff. The iterations are also terminated if

$$|p(z_{k+1})| = |p(z_k)| < 16nu|a_n|,$$

where the right hand side is a generous overestimate of the final roundoff made in computing $p(z)$ at the root of the smallest magnitude. The polynomial is then deflated as described in the previous section.

More details about this algorithm and methods for computing error bounds can be found in [14] and [15].

6.5.7 Laguerre's Method

In **Laguerre's method**[13] the polynomial $p(z)$ of degree n is approximated in the neighborhood of the point z_k by a special polynomial of the form

$$r(z) = a(z - w_1)(z - w_2)^{n-1},$$

where the parameters a, w_1 and w_2 are determined so that

$$p(z_k) = r(z_k), \qquad p'(z_k) = r'(z_k), \qquad p''(z_k) = r''(z_k). \tag{6.5.18}$$

If z_k is an approximation to a simple zero, α then the simple zero w_1 of $r(z)$ is taken as the new approximation z_{k+1} of α. Laguerre's method has very good global convergence properties for polynomial equations, and with *cubic* convergence for simple roots (real or complex). For multiple roots convergence is only linear.

In order to derive Laguerre's method we note that the logarithmic derivative of $p(z) = (z - \alpha_1) \cdots (z - \alpha_n)$ is

$$S_1(z) = \frac{p'(z)}{p(z)} = \sum_{i=1}^{n} \frac{1}{z - \alpha_i}.$$

Taking the derivative of this expression we obtain

$$-\frac{dS_1(z)}{dz} = S_2(z) = \left(\frac{p'(z)}{p(z)}\right)^2 - \frac{p''(z)}{p(z)} = \sum_{i=1}^{n} \frac{1}{(z - \alpha_i)^2}.$$

Using (6.5.18) to determine the parameters of the approximating polynomial r(z) we obtain the equations

$$S_1(z_k) = \frac{1}{z_k - w_1} + \frac{(n-1)}{z_k - w_2}, \qquad S_2(z_k) = \frac{1}{(z_k - w_1)^2} + \frac{(n-1)}{(z_k - w_2)^2}.$$

Eliminating $z_k - w_2$ gives a quadratic equation for the correction $z_k - w_1 = z_k - z_{k+1}$. After some algebra we obtain (check this!)

$$z_{k+1} = z_k - \frac{np(z_k)}{p'(z_k) \pm \sqrt{H(z_k)}}, \tag{6.5.19}$$

where

$$H(z_k) = (n-1)^2[p'(z_k)]^2 - n(n-1)p(z_k)p''(z_k).$$

The sign in the denominator in (6.5.19) should be chosen so that the magnitude of the correction $|z_{k+1} - z_k|$ becomes as small as possible.

For polynomial equations with only real roots, Laguerre's method is globally convergent, i.e., it converges for *every* choice of real initial estimate z_0. Suppose the roots are ordered such that $\alpha_1 \le \alpha_2 \le \cdots \le \alpha_n$. If $z_0 \in (\alpha_{j-1}, \alpha_j)$, $j = 2 : n$, then

[13] Edmund Nicolas Laguerre, 1834–1886, French mathematician at École Polytechnique, Paris and best known for his work on orthogonal polynomials.

Laguerre's method converges to one of the roots α_{j-1}, α_j; if $z_0 < \alpha_1$ or $z_0 > \alpha_n$ then convergence is to α_1 or α_n respectively.

For polynomial equations with complex roots, Laguerre's method no longer converges for every choice of initial estimate. However, experience has shown that the global convergence properties are good also in this case. In particular, if we take $z_0 = 0$, then Laguerre's method will usually converge to the root of smallest modulus. We finally remark that, as might be expected, for multiple roots convergence of Laguerre's method is only linear.

Consider the polynomial equation $p(z) = 0$ and assume that $a_n \neq 0$ so that $\alpha = 0$ is not a root. Now suppose that $a_{n-2}a_{n-1} \neq 0$, and take $z_0 = 0$ in Laguerre's method. A simple calculation gives

$$z_1 = \frac{-na_n}{a_{n-1} \pm \sqrt{H(z_0)}}, \qquad H(z_0) = (n-1)^2 a_{n-1}^2 - 2n(n-1)a_n a_{n-2}, \tag{6.5.20}$$

where the sign is to be chosen so the the $|z_1|$ is minimized. In particular, for $n = 2$, $H(z_0)$ is the discriminant of $p(z)$ and z_1 is the root of smallest modulus.

Example 6.5.4.

If there are complex roots, then there may be several distinct roots of smallest modulus. For example, the equation

$$p(z) = z^3 - 2z^2 + z - 2,$$

has roots $\pm i$ and 2. Using the above formula (6.5.20) for z_1 with $n = 3$, we get

$$z_1 = \frac{6}{1 \pm 2i\sqrt{11}} = \frac{2}{15} \pm i\frac{4\sqrt{11}}{15} = 0.06666666667 \pm 0.88443327743i.$$

Continuing the iterations with Newton's method we get convergence to one of the two roots $\pm i$,

$$z_2 = -0.00849761051 + 1.01435422762i, \; z_3 = -0.00011503062 + 1.00018804502i$$
$$z_4 = -0.00000002143 + 1.00000003279i, \; z_5 = -0.00000000000 + 1.00000000000i$$

Review Questions

1. Describe the method of iterated successive synthetic division for computing function values and derivatives of a polynomial.
2. Consider the polynomial $p(z) = z^4 - 2z^3 - 4z^2 + z + 1$. Using Descartes' rule of sign what can you deduce about the number of real positive roots?
3. Suppose that all roots of a polynomial equation are to be determined. Describe two methods to avoid the problem of repeatedly converging to roots already found.

4. Discuss the ill-conditioning of roots of polynomial equations. What famous polynomial did J. H. Wilkinson use as an example?

5. (a) What is the companion matrix of a polynomial $p(x) = x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n$?

(b) One approach to computing the eigenvalues of a matrix A is to find the coefficients of the characteristic polynomial $p_A(\lambda) = \det(\lambda I - A)$, and then solve the algebraic equation $p_A(\lambda) = 0$. Why should such a method usually be avoided?

6. What properties are satisfied by a Sturm sequence of real polynomials $p_0(x)$, $p_1(x), \ldots, p_m(x)$? Describe one way of generating a Sturm sequence using the Euclidian algorithm.

Problems and Computer Exercises

1. Apply Newton's method to determine one of the complex roots of the equation $z^2 + 1 = 0$. Start with $z_0 = 1 + i$.

2. Consider a polynomial with real coefficients

$$p(z) = a_0z^n + a_1z^{n-1} + \cdots + a_n, \quad a_i \neq 0, \quad i = 0 : n.$$

(a) Count the number of (real) additions and multiplications needed to compute a value $p(z_0)$ by synthetic division of $p(z)$ by $(z - z_0)$, when z_0 is a real and complex number, respectively.

(b) For a complex number $z_0 = x_0 + iy_0$, $p(z_0)$ can also be computed by performing the synthetic division of p(z) with the real quadratic factor

$$d(z) = (z - z_0)(z - \bar{z}_0) = z^2 - 2x_0z + (x_0^2 + y_0^2).$$

Derive a recursion for computing the quotient polynomial $q(z)$ and $p(z_0) = b_{n-1}z_0 + b_n$, where

$$\begin{aligned} q(z) &= b_0z^{n-2} + b_1z^{n-3} + \ldots + b_{n-2}, \\ p(z) &= q(z)d(z) + b_{n-1}z + b_n. \end{aligned}$$

Count the number of real additions and multiplications needed to compute $p(z_0)$ and also show how to compute $p'(z_0)$.

3. (a) Using the Cardano–Tartaglia formula the real root α to the equation $x^3 = x + 4$ can be written in the form

$$\alpha = \sqrt[3]{2 + \frac{1}{9}\sqrt{321}} + \sqrt[3]{2 - \frac{1}{9}\sqrt{321}}.$$

Use this expression to compute α. Discuss the loss of accuracy due to cancellation.

(b) Compute α to the same accuracy by Newton's method using the initial approximation $x_0 = 2$.

4. A method due to D. Bernouilli for obtaining roots of the algebraic equation

$$p(x) = x^n + a_1 x^{n-1} + \cdots + a_{n-1}x + a_n = 0$$

is based on the related linear difference equation

$$\mu_k + a_1\mu_{k-1} + \cdots + a_{n-1}\mu_{k-n+1} + a_n\mu_{k-n} = 0,$$

having the same coefficients as the algebraic equation.
(a) Show that the general solution to the difference equation is

$$\mu_k = c_1\alpha_1^k + c_2\alpha_2^k + \cdots + c_n\alpha_n^k,$$

where α_k, $k = 1 : n$, are the roots of the algebraic equation.
(b) Assume that the roots are ordered after decreasing magnitude and that $|\alpha_1| > |\alpha_2|$. Show that if α_1 is real, then for almost all choices of initial values $\mu_0, \ldots, \mu_{n-1}$, it holds that

$$\lim_{k\to\infty} \frac{\mu_k}{\mu_{k-1}} = \alpha_1.$$

(c) Let C be the companion matrix of $p(x)$. Show that the sequence μ_k can be generated by forming successive matrix-vector products

$$m_k = Cm_{k-1}, \quad m_k = \begin{pmatrix} \mu_{k+n-1} & \ldots & \mu_{k+1} & \mu_k \end{pmatrix}^T.$$

Show that $m_k = C^k m_0$.
Comment: When applied to a general matrix C this is known as the power method for computing eigenvalues.

5. In Graeffe's root-squaring method one separates even and odd powers of the polynomial $p(z)$ and squares the equation as follows

$$(a_0z^n + a_2z^{n-2} + a_4z^{n-4} + \cdots)^2 = (a_1z^{n-1} + a_3z^{n-3} + a_5z^{n-5} + \cdots)^2.$$

Putting $u = z^2$ the resulting equation becomes

$$q(u) = p(-z)p(z) = b_0u^n + b_1u^{n-1} + \cdots + b_n = 0.$$

This has roots equal to the squares of the roots of the original equation.
(a) Show that the coefficients b_k of $q(u)$ can be computed from

$$b_0 = a_0^2, \qquad (-1)^k b_k = a_k^2 + \sum_{j=1}^{k}(-1)^j 2a_{k-j}a_{k+j}, \quad k = 1, 2, \ldots, n.$$

(b) After squaring m times we obtain (after normalizing $A_0 = 1$) an equation in $u = z^{2m}$

$$u^n + A_1u^{n-1} + A_2u^{n-2} + \cdots + A_n = 0,$$

with roots $\beta_k = \alpha_k^{2m}$. Assume that the roots α_k of the original equation $p(z) = 0$ are real and distinct. Use the relations between coefficients and roots of an algebraic equation to show that for m large enough we have

$$\beta_1 \approx -A_1, \qquad \beta_2 \approx -A_2/A_1, \qquad \beta_3 \approx -A_3/A_2, \ldots.$$

(c) Square the polynomial $z^3 - 8z^2 + 17z - 10$ three times, and then use the relations in (b) to compute approximations to its three real roots.

6. Consider the iteration $z_{n+1} = z_n^2 + c$, where $c = p + iq$ is a fixed complex number. For a given z_0 the sequence of iterates $z_n = x_n + iy_n$, $n = 0, 1, 2, \ldots$ may either converge to one of the two roots of the quadratic equation $z^2 - z + c = 0$ or diverge to infinity. Consider z_0 chosen, e.g., in the unit square of the complex plane. The boundary separating the region of convergence from other points in the plane is a very complex **fractal curve** know as the **Julia set**. The **Mandelbrot set** is obtained by fixing $z_0 = 0$ and sweeping over values of c in a region of the complex plane.

(a) Picture the Julia set as follows. Set $c = 0.27334 + 0.000742i$. Sweep over points of z_0 in the region $-1 \leq \Re z_0 \leq 1$, $-1.3 \leq \Im z_0 \leq 1.3$. If $|z_N| < R$, for $N = 100$ and $R = 10$ color the point z_0 black. otherwise color the point from hot (red) to cool (blue) according to how fast the iteration is diverging, i.e. according to how fast the inequality $|z_n| > R$ becomes satisfied.

(b) Picture the Mandelbrot set in a similar way. Sweep over values of c in the region $-2.25 \leq \Re c \leq 0.75$, $-1.5 \leq \Im c \leq 1.5$.

Notes and References

An interesting historical account of Newton's method is given in Ypma [30]. Newton's method is contained in his book "Method of Fluxions" written 1671, but not published until 1736. Joseph Raphson was allowed to see Newton's work and Newton's method first was first published in a book by Raphson 1690. This is why the method in English literature is often called the Newton–Raphson method. The first to give a modern description of Newton's method using derivatives seems to have been Thomas Simpson 1740. Edmund Halley was contemporary with Isaac Newton and his third order method was published more than 300 years ago [8].

Halley's method has been rediscovered by J. H. Lambert [13] and numerous other people; see Traub [25, Sec. 5.22]. A nice exposition of this and other third order methods is given by Gander [6]. Families of iteration methods of arbitrary order ar studied in a remarkable paper by Schröder [22]. An English translation of this paper is given by G. W. Stewart [23]. The determinant family of iteration functions $B_p(x)$ is a special case of a parametrized family of iteration functions for polynomials given by Traub [26]; see also [10, Sec. 4.4]. This family was derived independently by Kalantari et al. [11].

One of the best algorithms to combine bisection and interpolation was developed by van Wijngaarden and Dekker at Mathematical Center in Amsterdam in the 1960s; see [3]. It was taken up and improved by Brent [2]; see also [5], Section 7.2. Brent's new algorithm, in contrast to Dekker's, never converges much more slowly than bisection. Fortran and C versions of some of the zero-finding and minimization routines given in the book are available from Netlib.

Several comprehensive monographs dealing with methods for solving scalar nonlinear equations are available. Traub [25, 1964] gives an exhaustive enumeration of iteration methods with and without memory, with their order of convergence and efficiency index. Much classical material is also found in Ostrowski [19, 1973]. The elegant treatment by Householder [10, 1970] also deserves special mention.

The recently reprinted book by Brent [2] deals exclusively with methods which only uses function values for finding zeros and minima of functions of a single variable. It is unique in the careful treatment of algorithmic details that are crucial when developing reliable computer codes.

There is is a vast literature on methods for solving algebraic equations. An excellent introduction is given by Householder [10] and detailed surveys found, e.g., in Durand [4] and Sendov et al. [24]. The modified Newton method is due to Madsen [14] Another much used method for computing polynomial roots is the Jenkins–Traub method, which is included in the IMSL library. A good discussion of this rather complex method is found in Ralston and Rabinowitz [21, Sec. 8.11]. An evaluation of the speed and accuracy of the QR algorithm, used in Matlab has been given by [7].

The theory of Sturm sequences are treated in [10, Sec. 2.5]. The quadtree method was used by Weyl [27] to give a constructive proof of the fundamental theorem of algebra. An interesting analysis of the efficient implementation of this method is given by Pan [20], who also gives a brief account of the history of algorithms for solving polynomial equations.

[1] David H. Bailey, Jon M. Borwein, Peter B. Borwein, and Simon Plouffe. The quest for pi. *Notes Amer. Math. Soc.*, 19:1:50–57, 1975.

[2] Richard P. Brent. *Algorithms for Minimization without Derivatives.* Prentice-Hall, Englewood Cliffs, NJ, 1973. Reprinted by Dover Publications, Mineola, New York, 2002.

[3] J. C. P. Bus and T. J. Dekker. Two efficient algorithms with guaranteed convergence for finding a zero of a function. *ACM Trans. Math. Software*, 1:330–345, 1975.

[4] E. Durand. *Solutions numériques des équations algébriques. Tom I: Équations du type $F(x) = 0$, racines d'un polynôme.* Masson, Paris, 1960.

[5] George E. Forsythe, Michael A. Malcolm, and Cleve B. Moler. *Computer Methods for Mathematical Computations.* Prentice-Hall, Englewood Cliffs, NJ, 1977.

[6] Walter Gander. On Halley's iteration method. *American Mathematics Monthly*, 92:131–134, 1985.

[7] S. Goedecker. Remark on algorithms to find roots of polynomials. *SIAM J. Sci. Comput.*, 15:1059–1063, 1994.

[8] Edmond Halley. A new and general method of finding the roots of equations. *Philos. Trans. Roy. Soc. London*, 18:136, 1694.

[9] Peter Henrici. *Applied and Computational Complex Analysis. Volume 1 Power Series, Integration, Conformal Mapping, Location of Zeros.* Wiley Classics Library, New York, 1988. Reprint of the 1974 original.

[10] Alston S. Householder. *The Numerical Treatment of a Single Nonlinear Equation.* McGraw-Hill, New York, 1970.

[11] Bahman Kalantari, Iraj Kalantari, and Rahim Zaare-Nahandi. A basic family of iteration functions for polynomial root finding and its characterizations. *J. Comput. Appl. Math.*, 80:209–226, 1997.

[12] Göran Kjellberg. Two observations on Durand–Kerner's root finding method. *BIT*, 24:556–559, 1973.

[13] Johann H. Lambert. *Beiträge zum Gebrauche der Mathematik und deren Anwendungen. Zweiter Theil.* 1770.

[14] Kaj Madsen. A root-finding algorithm based on Newton's method. *BIT*, 13:71–75, 1973.

[15] Kaj Madsen and John K. Reid. Fortran subroutines for finding polynomial zeros. Tech. Report A.E.R.E. R.7986, Computer Science and Systems Division, A.E.R.E. Harwell, 1975.

[16] Cleve B. Moler. *Numerical Computing with Matlab.* SIAM, Philadelphia, PA, 2004.

[17] Ramon E. Moore. *Interval Analysis.* Prentice-Hall, Englewood Cliffs, NJ, 1966.

[18] James M. Ortega and Werner C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables.* Academic Press, New York, 1970.

[19] A. M. Ostrowski. *Solution of Equations in Euclidian and Banach Spaces.* Academic Press, New York, third edition, 1973.

[20] V. Y. Pan. Solving a polynomial equation: Some history and recent progress. *SIAM Review*, 39:187–220, 1997.

[21] Anthony Ralston and Philip Rabinowitz. *A First Course in Numerical Analysis.* McGraw-Hill, New York, second edition, 1978.

[22] Ernst Schröder. Über unendlich viele Algorithmen zur Auflösung der Gleichungen. *Mathematische Annalen*, 2:317–365, 1870.

[23] Ernst Schröder. On infinitely many algorithms for solving equations (translated by G. W. Stewart). Tech. Report TR-92-121, Department of Computer Science, University of Maryland, College Park, MD, 1992.

[24] Bl. Sendov, A. Andreev, and N. Kjurkchiev. Numerical solution of polynomial equations. In P. G. Ciarlet and J. L. Lions, editors, *Handbook of Numerical Analysis*, volume III, pages 629–778. Elsevier Science, Cambridge, UK, 1994.

[25] J. F. Traub. *Iterative Methods for the Solution of Equations.* Prentice-Hall, Englewood Cliffs, NJ, 1964.

[26] J. F. Traub. A class of globally convergent iteration functions for the solution of polynomial equations. *Math. Comp.*, 20:113–138, 1966.

[27] Hermann Weyl. Randbemerkungen zu Hauptproblemen der Mathematik, II, Fundamentalsatz der Algebra und Grundlagen der Mathematik. *Math. Z.*, 20:131–151, 1924.

[28] James H Wilkinson. The perfidious polynomial. In G. H. Golub, editor, *Studies in Numerical Analysis*, pages 1–28. American Mathematical Society, Providence, RI, 1984.

[29] L. Yao and Adi Ben-Israel. The Newton and Halley methods for complex roots. *Amer. Math. Monthly*, 105:806–818, 1998.

[30] Tjalling J. Ypma. Development of the Newton–Raphson method. *SIAM Review*, 37:531–551, 1995.

Index