Chapter 6
Solving Scalar Nonlinear Equations

If we start with an approximation to a zero that is appreciably more correct than the limiting accuracy, a single (Newton) iteration will usually spoil this very good approximation and produce one with an error which is typically of the limiting accuracy.
—J. H. Wilkinson

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 several 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, for example, for the reduced cubic equation), this is often so complicated that using Newton's method is much 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 it suffices that the function $f(x)$, and preferably some of its derivatives, can be evaluated for given numerical values of x for numerical methods to be applicable.

It is not uncommon in applications that each function value is obtained by a complicated computation, for example, 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 limiting by *any numerical method*. The effect of such rounding errors depends on the conditioning of the roots and is discussed in Sec. 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 Sec. 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 use of more 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 Sec. 6.3. A classical problem is that of determining all real or complex roots of an algebraic equation. Special features and methods for this problem are taken up in Sec. 6.5.

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

6.1.2 The Bisection Method

It is often advisable to start by 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 6.1.1 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 (*Intermediate Value Theorem*).

Assume that the function $f(x)$ is continuous for $x \in [a, b]$, $f(a) \neq f(b)$, 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).

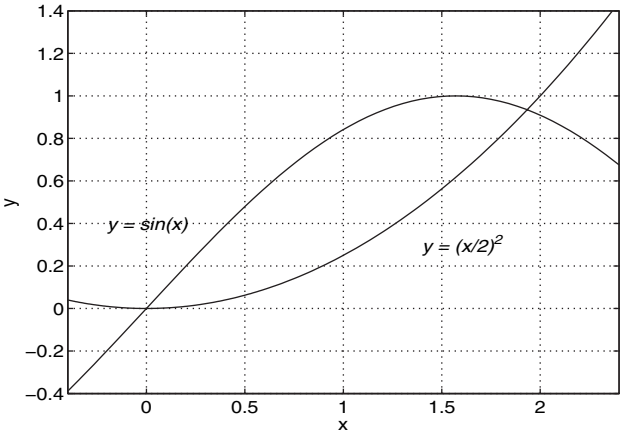

Figure 6.1.1. *Graph of curves* $y = (x/2)^2$ *and* $\sin x$.

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

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

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

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

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 and Example 6.1.3).

We can assume that $f(m_k) \neq 0$, since otherwise we have found a root. The new interval is then determined by

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

From the construction it follows immediately that $f(a_{k+1})f(b_{k+1}) < 0$ (see also Figure 6.1.1) and therefore the interval I_{k+1} also contains a root of $f(x) = 0$.

If the sign of $f(m_k)$ has been correctly evaluated for $k = 1 : n$, then 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 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.

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 $0.2 \cdot 2^{-6} = 0.003125$ (see Figure 6.1.2).

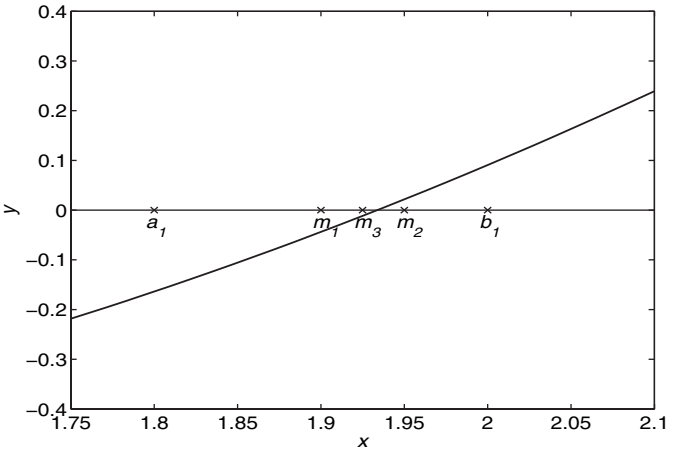

Figure 6.1.2. *The bisection method.*

Example 6.1.3.

The inequalities $a \leq fl(\frac{1}{2}(a+b)) \leq b$, where a and b are floating-point numbers with $a \leq b$, can be violated in floating-point arithmetic. For example, assume that base 10 arithmetic with six digits is used. Taking $a = 0.742531$ and $b = 0.742533$ we obtain $fl(a+b) = 1.48506$ (rounded) and $\frac{1}{2}(a+b) = 0.742530 < a$.

On the other hand the inequalities $a \leq fl(a + \frac{1}{2}(b-a)) \leq b$ hold in floating-point arithmetic, for any base β. (Recall that by Lemma 2.3.2 $fl(b-a)$ is evaluated *exactly* in binary arithmetic if $b/2 \leq a \leq 2b$.) With a and b as given, we get the correct value 0.742532.

An algorithmic description of the bisection method is given below. Let f be a given function and $I = [a, b]$ an interval such that $b > a$ and $f(a)f(b) \leq 0$. The bisection algorithm attempts to compute an approximation to a root $m \in I$ of $f(x) = 0$, with an error less than a specified tolerance $\tau > 0$. Note that the given 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 too small (e.g., smaller than the spacing between the floating-point numbers between a and b).

ALGORITHM 6.1. *Bisection.*

The function `bisect` computes an approximation r to a local root of a given function in the interval $I = (a, b)$ with an error less than a specified tolerance $\tau + u \max(|a|, |a|)$, where u is the unit roundoff.

```
function root = bisect(fname,a,b,tau);
%
% [a,b] is an initial interval such that
% f(a)*f(b) < 0.
fa = feval(fname,a);
fb = feval(fname,b);
while abs(b-a) > tau + eps*max(abs(a),abs(b))
   mid = a + (b - a)/2;
   fmid = feval(fname,mid);
   if fa*fmid <= 0
   % Keep left endpoint a
      b = mid;
   else
   % Keep right endpoint b
      a = mid; fa = fmid;
end;
root = a  + (b - a)/2;
end;
```

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 Sec. 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, for example, functions that are 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), and in the 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 step length h and in the first **hunting phase** compute successively function values $f(a+h)$, $f(a+2h)$, $f(a+4h)$, ..., 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 **multisection 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

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

multisections of order p. Thus, the efficiency of multisection 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 multisection may perform better if the subintervals can be processed in parallel.

There are several other applications of the bisection algorithm. For example, in Sec. 4.4.3 we considered evaluating the nonzero B-splines for a given argument x. Then we first have to search an ordered sequence of knots $\tau_0, \ldots, \tau_m$ 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*, for example, 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 Limiting Accuracy and Termination Criteria

In the following we denote by $\overline{f}(x)$ 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 6.1.3. 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 $[a, b]$!

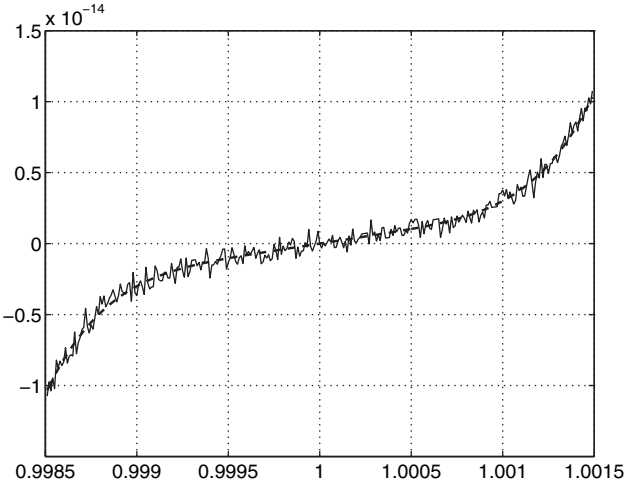

Figure 6.1.3. *Limited-precision approximation of a continuous function.*

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, for example, Ostrowski [279, Chapter 2].

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 a **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

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

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 **limiting 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 $f(x) = \sin x - (x/2)^2$. Now $f'(x) = \cos x - x/2$ and it is easily verified that

$$|f'(x)| > 1.31 = m_1, \quad x \in [1.93, 1.94].$$

Further, using six decimals we get $\sin 1.93375 = 0.934852 \pm 0.5 \cdot 10^{-6}$, and $(x/2)^2 = 0.966875^2 = 0.934847 \pm 0.5 \cdot 10^{-6}$. Then (6.1.6) gives 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 disk $K = \{z \mid |z - z_0| \leq \eta\}$ for some $\eta > 0$. If $|f'(z)| \geq m_1$ in K and $|f(z_0)| \leq \eta m_1$, 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

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

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

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

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

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

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 also be expected when there are several

distinct but "close" roots. An instructive example is the Wilkinson polynomial, studied in Sec. 6.5.2.

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

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

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 $0.99992929 \le x \le 1.0000707$, 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, i.e., with a relative error equal to the *square root of the machine precision*.

An important practical question concerning iterative methods is how to stop them. There are a few different **termination criteria** in practical use. The simplest is to stop after a preset number of iterations. This is, in general, too crude, but it is advisable always to specify the maximum number of iterations allowed, to guard against unforeseen problems. For example, programming errors could otherwise lead to an infinite loop being executed.

If the iteration method produces a sequence of bracketing intervals $[a_k, b_k]$ the iterations can be terminated when

$$|b_k - a_k| \le \text{tol}, \quad \text{tol} = \tau + 2u|x_n|, \tag{6.1.11}$$

where $\tau > 0$ is a user specified absolute tolerance and u is the rounding unit (see Sec. 2.2.2). The second term assures that the iterations are terminated when the roundoff level of the floating-point arithmetic is reached.

If for a simple root a lower bound for $|f'(\alpha)|$ is known, the iterations can be terminated on the basis of the error estimate (6.1.7). But it is usually more effective to iterate a few extra times, rather than make the effort to use a special formula for error estimation.

The termination criteria must be able to deal with the possibility that the user specified tolerance is too small (or the root too ill-conditioned) and cannot be attained. In this case from some step onward 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 limiting accuracy. If we are using a bracketing method like the bisection method, the iterations will continue until the criterion (6.1.11) is satisfied. This, of course, does *not* ensure that the root actually has been determined to this precision.

For iterative methods with fast convergence (like Newton's method) the following termination criterion can often be used: Stop when for the first time the approximation x_n satisfies the two conditions

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

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 limiting 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 limiting accuracy is obtained may be small, even if the user specified tolerance is much larger than ϵ_α.

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

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

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.

Example 6.1.6.

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

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

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 6.1.4. The convergence is slow and we get $x_9 = 0.5641$ (correct value 0.567143).

As was shown already in Sec. 1.1.1 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, then α is called a **point of attraction**; otherwise α is a **point of repulsion**. The case $|\phi'(\alpha)| = 1$ is indeterminate; the fixed-point iteration $x_{n+1} = \phi(x_n)$ can either converge or diverge (see Problem 6.1.10).

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. The following important theorem not only provides a solid basis for iterative numerical techniques, but also is an

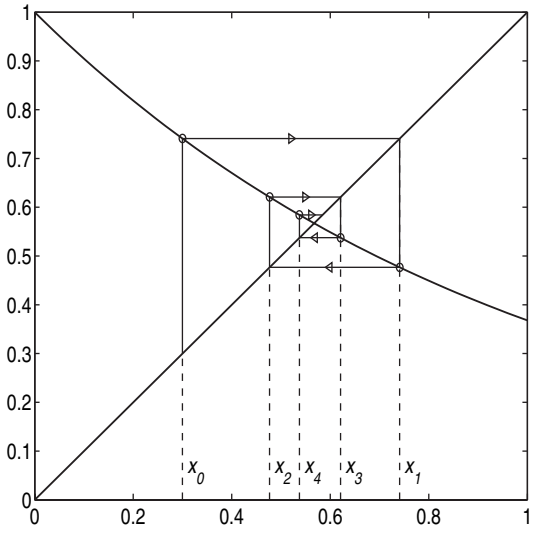

Figure 6.1.4. *The fixed-point iteration* $x_{k+1} = e^{-x_k}$, $x_0 = 0.3$.

important tool in theoretical analysis. Note that *the existence of a fixed point is not assumed a priori*.

Theorem 6.1.5 (*Contraction Mapping Theorem*).

Let x_0 *be a starting point, and consider the iteration* $x_{n+1} = \phi(x_n)$, $n = 1, 2, \ldots$. *Let*

$$J_\rho = \{x \mid |x - x_0| < \rho\}$$

be an open interval of length ρ *around* x_0 *and assume that* $x \to \phi(x)$ *is a* **contraction mapping**, *i.e., for arbitrary points* s *and* t *in* J_ρ

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

Then if

$$|x_0 - x_1| \le (1 - L)\rho, \tag{6.1.17}$$

the equation $x = \phi(x)$ *has a unique solution* α *in the closed interval* $|x - x_0| \le \rho$. *This solution can be obtained by the convergent iteration process* $x_{k+1} = \phi(x_k)$, $k = 0, 1, \ldots$. *We have the error estimate*

$$|x_k - \alpha| \le |x_k - x_{k-1}|\frac{L}{1 - L} \le |x_1 - x_0|\frac{L^k}{1 - L}. \tag{6.1.18}$$

Proof. We first prove the uniqueness. If there were two solutions x' and x'', we would get $x' - x'' = \phi(x') - \phi(x'')$ so that

$$|x' - x''| = |\phi(x') - \phi(x'')| \le L|x' - x''|.$$

Since $L < 1$, it follows that $|x' - x''| = 0$, i.e., $x' = x''$.

By (6.1.17) we have $|x_1 - x_0| = |\phi(x_0) - x_0| \le (1 - L)\rho$, and hence $x_1 \in J_\rho$. We now use induction to prove that $x_n \in J_\rho$ for $n < j$, and that

$$|x_j - x_{j-1}| \le L^{j-1}(1 - L)\rho, \qquad |x_j - x_0| \le (1 - L^j)\rho.$$

We already know that these estimates are true for $j = 1$. Using the triangle inequality and (6.1.16) we get

$$\begin{aligned} |x_{j+1} - x_j| &= |\phi(x_j) - \phi(x_{j-1})| \le L|x_j - x_{j-1}| \le L^j(1 - L)\rho, \\ |x_{j+1} - x_0| &\le |x_{j+1} - x_j| + |x_j - x_0| \le L^j(1 - L)\rho + (1 - L^j)\rho \\ &= (1 - L^{j+1})\rho. \end{aligned}$$

This proves the induction step, and it follows that the sequence $\{x_k\}_{k=0}^{\infty}$ stays in J_ρ. We also have for $p > 0$

$$\begin{aligned} |x_{j+p} - x_j| &\le |x_{j+p} - x_{j+p-1}| + \cdots + |x_{j+1} - x_j| \\ &\le (L^{j+p-1} + \cdots + L^j)(1 - L)\rho \le L^j(1 - L^p)\rho \le L^j\rho, \end{aligned}$$

and hence $\lim_{j\to\infty} |x_{j+p} - x_j| = 0$. The sequence $\{x_k\}_{k=0}^{\infty}$ therefore is a Cauchy sequence, and since $\mathbf{R}^n$ is complete has a limit α. Since $x_j \in J_\rho$ for all j it follows that $\alpha \in \{x \mid |x - x_0| \le \rho\}$.

Finally, by (6.1.16) ϕ is continuous, and it follows that $\lim_{k\to\infty} \phi(x_k) = \phi(\alpha) = \alpha$. The demonstration of the error estimates (6.1.18) is left as an exercise for the reader. □

There is an analogue of Theorem 6.1.5 for complex functions $\phi(z)$ analytic in a circle $K = \{z \mid |z - \alpha| \le \rho\}$ containing the initial approximation z_0. Indeed, Theorem 6.1.5 holds in a much more general setting, where $\phi : S_r \to \mathcal{B}$, and $\mathcal{B}$ is a Banach space.[178] The proof goes through with simple modifications. In this form the theorem can be used to prove existence and uniqueness for initial value problems for ordinary differential equations.

An estimate of the error in x_n, which depends only on x_0, x_1 and the Lipschitz constant L, 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 $|x_{i+1} - x_i| \le C^i|x_1 - x_0|$, and hence

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

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

$$|\alpha - x_n| \le \frac{1}{1 - C}|x_{n+1} - x_n| \le \frac{C^n}{1 - C}|x_1 - x_0|. \tag{6.1.19}$$

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!

[178] Recall that a Banach space is a normed vector space which is complete, i.e., every Cauchy sequence converges to a point in $\mathcal{B}$; see Dieudonné [100].

For a linearly convergent fixed-point iteration the sequence $\{x_j - \alpha\}$ approximately forms a geometric series. Then, as seen in Sec. 3.4.2, a more rapidly convergent sequence $\{x'_j\}$ can be obtained by Aitken extrapolation,

$$x'_j = x_j - (\nabla x_j)^2/\nabla^2 x_j. \tag{6.1.20}$$

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

Example 6.1.7.

Let $\phi(x) = 1 - \frac{1}{2}x^2$, and consider the fixed-point iteration $x_{n+1} = \phi(x_n)$, with $x_0 = 0.8$, $n = 0, 1, 2, \ldots$. In this example the theoretical criterion (3.4.2) is satisfied, since one can show that $x_n \to a = \sqrt{3} - 1 \approx 0.73205$,

$$\nabla x_n/\nabla x_{n-1} \to \phi'(a) = -a, \quad n \to \infty.$$

(See the discussion of iteration in Sec. 1.1.1.)

We obtain $x_1 = 0.68$, $x_2 = 0.7688$, and $x'_2 = 0.7688 - (0.0888)^2/0.2088 = 0.73103$. The error in x'_2 is only about 3% of the error in x_2. Iterated Aitken yields, with $u = 2^{-53} = 1.1 \cdot 10^{-16}$, the results $x'_2 = 0.73103\ldots$, $x''_4 = 0.73211$, etc., and the errors

$$e'_2 = -10 \cdot 10^{-3}, \quad e''_4 = 5.6 \cdot 10^{-5}, \quad e'''_6 = 7.8 \cdot 10^{-7}, \quad e^{(4)}_8 = -2.8 \cdot 10^{-9},$$

after 2, 4, 6, 8 evaluations of the function ϕ, respectively. We shall see (Problem 6.1.13) that the accelerated sequence may converge, even if the basic iteration $x_{n+1} = \phi(x_n)$ diverges.

When the basic sequence $\{x_n\}$, as in the above example, is produced by a convergent iterative process, one can apply Aitken acceleration in a different, usually more efficient way. This can be called **active Aitken acceleration**, since the result of an acceleration is actively used in the basic iterative process. We start as before by computing $x_1 = \phi(x_0)$, $x_2 = \phi(x_1)$ and apply (6.1.20) 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

$$z_{n+1} = \psi(z_n), \quad \psi(z) = z - \frac{(\phi(z) - z)^2}{(\phi(\phi(z)) - \phi(z)) - (\phi(z) - z)}. \tag{6.1.21}$$

This is repeated until some termination criterion is satisfied.

It can be shown that active Aitken extrapolation applied to the fixed point iteration $x_{n+1} = \phi(x_n)$ is equivalent to Steffensen's method applied to the equation $f(x) = x - \phi(x) = 0$; see (6.2.10).

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 rapidly converging fixed-point iteration. Basic concepts to quantify the rate of convergence will now be introduced.

Definition 6.1.6.

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* $0 < C < \infty$,

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

where q *need not be an integer.* 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, *respectively.*

More precisely, the order q in Definition 6.1.6 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 $C = 0$, 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 [278, Chap. 9] and Brent [44, 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 the **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. But the limit in (6.1.22) does not exist if $q = p$, is zero if $q < p$, and is 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 of 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 is $\delta_n = -\log_{10}|x_n - \alpha|$. 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 $\epsilon_{n+1} \leq C\epsilon_n^2$, it follows that ϵ_n, for $n = 2, 3, \ldots$, is bounded by

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

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

Definition 6.1.7.

Consider an iteration method with convergence order $q \geq 1$. 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

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

is the **efficiency index** *of the iteration.*

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

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

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

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

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

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

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

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

6.1.1 What limits the final accuracy of a root computed by the bisection algorithm?

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

(b) Assume 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?

6.1.3 (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.

6.1.4 (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.

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

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

6.1.6 What two quantities determine the limiting accuracy of a simple root α to the equation $f(x) = 0$? Give an example of an ill-conditioned root.

6.1.7 Discuss the choice of termination criteria for iterative methods.

Problems and Computer Exercises

6.1.1 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) $3x^2 + \tan x$.

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

6.1.3 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.

6.1.4 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) $e^{-2x} = 1 - x$.

6.1.5 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?

6.1.6 Let k be a given nonnegative 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.

6.1.7 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 the *geometric* mean

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

then the worst case *relative* error is minimized. Do Example 6.1.2 using this variation of the bisection method.

6.1.8 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.

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

6.1.10 (Wittmeyer-Koch)

(a) Show that the iteration $x_{n+1} = \phi(x_n)$, where $\phi(x) = x + (x-1)^2$, has a fixed point $\alpha = 1$, and $|\phi'(\alpha)| = 1$. Then verify that it converges by graphing the iteration for $x_0 = 0.6$.

(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.

6.1.11 (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 $x_0 < 1$.

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

6.1.13 Apply active Aitken acceleration to the iteration formula $s_{n+1} = \phi(s_n)$, where $\phi(s) = 1 - \frac{1}{2}s^2$, until either you have 10 correct decimals, or there are clear indications that the process is divergent.

(a) with $s_0 = 0.8$; (b) with $s_0 = -2.7$.

6.2 Methods Based on Interpolation

6.2.1 Method of False Position

Given two initial approximations $a_0 = a$ and $b_0 = b$ such that $f(a)f(b) < 0$, a nested sequence of intervals $(a_0, b_0) \supset (a_1, b_1) \supset (a_2, b_2) \supset \cdots$ such that $f(a_n)f(b_n) < 0$, $n = 0, 1, 2, \ldots$, can be generated as follows. Given (a_n, b_n), we take x_{n+1} to be the intersection of the x-axis and 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

$$x_{n+1} = a_n - f(a_n)\frac{a_n - b_n}{f(a_n) - f(b_n)}, n = 0, 1, 2, \ldots. \tag{6.2.1}$$

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 $a_{n+1} = a_n$. As for bisection, convergence to a root is guaranteed (in exact arithmetic) for a continuous function $f(x)$. This is the **false-position method** or, in Latin, **regula falsi**.[179]

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 $[a, b]$.

[179] Regula falsi is a very old method that originated in fifth century Indian texts and was used in medieval Arabic mathematics. It got its current name from the Italian mathematician Leonardo Pisano, also known as Leonardo Fibonacci (ca 1170–1250). He is considered to be one of the most talented mathematicians of the Middle Ages.

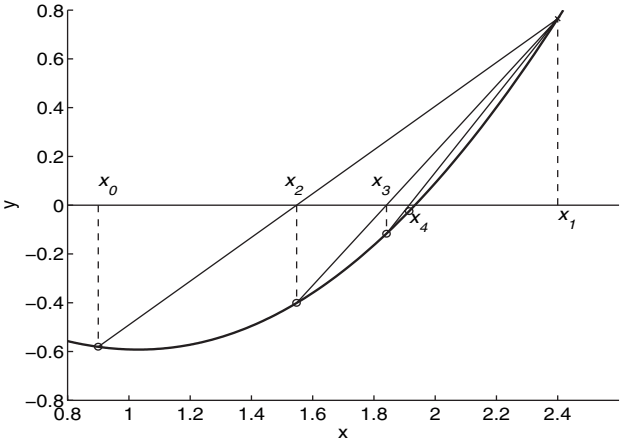

Figure 6.2.1. *The false-position method.*

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 as follows.

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 to 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.

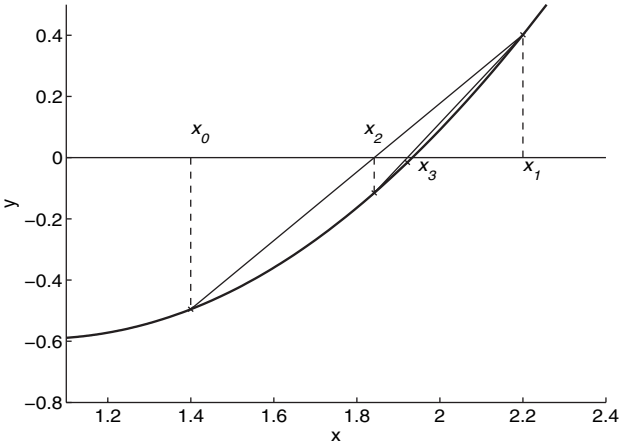

Figure 6.2.2. *The secant method.*

Given initial approximations $x_{-1} = a$ and $x_0 = b$, approximations $x_1, x_2, , x_3, \ldots$ 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.

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 Sec. 6.1.4. Sometimes methods of this form, which use old information at points $x_{n-k}, k \geq 1$, 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} are both very close to the root α and not bracketing α, then the resulting rounding error in x_{n+1} can 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 roundoff error in x_{n+1} comes from the error in $f(x_n)$. Note that (6.2.3) should *not* be "simplified" to 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 shown in the following table.

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. But 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 $\zeta_n \in \text{int}(x, x_{n-1}, x_n)$ and

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

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

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

But *in the secant method there is no guarantee that the computed sequence of approximations stays 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 $x_0, x_1, x_2, \ldots, x_n$, we have

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

Denote by ϵ_n the error in the nth iterate of the secant iteration Let q be the unique positive root of the equation $\mu^2 - \mu - 1 = 0$ and set

$$K = \max\left(M|\epsilon_0|, (M|\epsilon_1|)^{1/q}\right). \tag{6.2.7}$$

Then it holds that

$$|\epsilon_n| \le \frac{1}{M}K^{q^n}, \quad n = 0, 1, 2, \ldots; \tag{6.2.8}$$

i.e., the secant iteration has convergence order equal to $q = (1 + \sqrt{5})/2 = 1.618\ldots$.

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

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

To compare the efficiency of the secant method and Newton's method, which is quadratically convergent, we use the efficiency index introduced in Sec. 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

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

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,

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

i.e., *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 ratio

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

will have constant sign for all n. Then if $\alpha \in \operatorname{int}(x_0, x_1)$ and $\epsilon_0\epsilon_1 < 0$, 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 Interpolation Methods

The secant method does not achieve quadratic convergence. Indeed, it can be shown that under very weak restrictions no iterative method using only one function evaluation per step can achieve this. In **Steffensen's method**

$$x_{n+1} = x_n - \frac{f(x_n)}{g(x_n)}, \quad g(x_n) = \frac{f(x_n + f(x_n)) - f(x_n)}{f(x_n)}, \tag{6.2.10}$$

two function evaluations are used. This method can be viewed as a variant of the secant method, where the step size used to approximate the first derivative goes to zero as $f(x)$.

To show quadratic convergence we put $\beta_n = f(x_n)$ and expand $g(x_n)$ in a Taylor series about x_n; we get

$$g(x_n) = f'(x_n)\left(1 - \frac{1}{2}h_n f''(x_n) + O(\beta_n^2)\right),$$

where $h_n = -f(x_n)/f'(x_n)$ is the Newton correction. Thus,

$$x_{n+1} = x_n + h_n\left(1 + \frac{1}{2}h_n f''(x_n) + O(\beta_n^2)\right).$$

Using the error equation for Newton's method we get

$$\frac{\epsilon_{n+1}}{\epsilon_n^2} \;\to\; \frac{1}{2}\frac{f''(\alpha)}{f'(\alpha)}\big(1 + f'(\alpha)\big),$$

where $\epsilon_n = x_n - \alpha$. This shows that Steffenson's method is of second order.

Steffensen's method is of particular interest for solving *systems* of nonlinear equations. For a generalization to nonlinear operator equations on a Banach space, see [213].

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 [354, pp. 67–75]):

$$\epsilon_{n+1} = -\frac{f^{(r+1)}(\zeta_n)}{(r+1)!\,p'(\eta_n)} \prod_{i=0}^{r} \epsilon_{n-i}, \tag{6.2.11}$$

where $\zeta_n \in \text{int}\,(\alpha, x_{n-1}, x_n)$ and $\eta_n \in \text{int}\,(\alpha, x_{n+1})$. (Recall that by $\text{int}\,(a, b, \ldots, w)$ we denote the smallest interval that contains the points $a, b, \ldots, w$.) In the special case $r = 2$ we get the quadratic equation

$$p(x) = f_n + (x - x_n)[x_n, x_{n-1}]f + (x - x_n)(x - x_{n-1})[x_n, x_{n-1}, x_{n-2}]f. \tag{6.2.12}$$

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

$$h_n^2[x_n, x_{n-1}, x_{n-2}]f + \omega h_n + f_n = 0, \tag{6.2.13}$$

where

$$\omega = [x_n, x_{n-1}]f + (x_n - x_{n-1})[x_n, x_{n-1}, x_{n-2}]f. \tag{6.2.14}$$

The root closest to x_n corresponds to the root h_n of smallest absolute value to (6.2.13). 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

$$x_{n+1} = x_n + h_n, \quad h_n = -\frac{2f_n}{\omega \pm \sqrt{\omega^2 - 4f_n\,[x_n, x_{n-1}, x_{n-2}]f}}, \tag{6.2.15}$$

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 (6.2.13) 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 (6.2.11) it follows that

$$\epsilon_{n+1} = -\frac{f'''(\zeta_n)}{3!\,p'(\eta_n)}\epsilon_n\epsilon_{n-1}\epsilon_n. \tag{6.2.16}$$

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.

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

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

(cf. Sec. 4.2.2). 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, $x^i = x_n + F(x^{i-1})$, $i = 1, 2, \ldots$, 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

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

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

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

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

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 $x_{n+1} = q(0)$, i.e.,

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

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 6.2.3 and 6.2.4.) The case $r = 2$ corresponds to inverse quadratic interpolation:

$$x_{n+1} = x_n - y_n \, [y_n, y_{n-1}]g + y_n y_{n-1} \, [y_n, y_{n-1}, y_{n-2}]g. \tag{6.2.17}$$

Note that it has to be required that y_n, y_{n-1}, and y_{n-2} are distinct. This method has the same order of convergence as the Muller–Traub method.

Even if this is the case it is not always safe to compute x_{n+1} from (6.2.17). 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 (6.2.17) 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 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; see [61]. It uses a rapidly convergent method for smooth functions and the slow but sure bisection method when necessary. This

algorithm was taken up and improved by Brent [44]; see also [123, Section 7.2]. In contrast to Dekker's algorithm, Brent's new algorithm `zeroin` never converges much more slowly than bisection.

In each of the iterations of `zeroin` three approximations a, b, and c to the root are present. Initially we are given an interval $[a, b]$ such that $f(a)$ and $f(b)$ have opposite sign, and set $c = a$. In the following steps a, b, and c are arranged so that

- $f(a)$ and $f(b)$ have opposite sign, and $|f(b)| \leq |f(a)|$.
- b is the latest iteration.
- c is the value of b in the previous step.

At all times a and b bracket the zero and the length $|b - a|$ of the interval shrinks in each iteration.

If $c = a$ a secant step is taken; otherwise inverse quadratic interpolation is used. If the computed step gives an approximation in $[a, b]$ (and not too close to one of the endpoints) take it; otherwise a bisection step is taken. The iterations are terminated if $f(b) = 0$, or

$$|b - a| \leq \text{tol} + 4u|b|,$$

where tol is a user tolerance and u the unit roundoff. The midpoint $r = b + 0.5(a - b)$ is then returned as the root.

Brent claims that his version of `zeroin` will always converge, even in floating-point arithmetic. Further, if f has a continuous second derivative the method will eventually converge at least as fast as the secant method. In this case typically about ten function evaluations are needed. He never found a function requiring more than three times the number of evaluations needed for bisection. On average, using inverse quadratic interpolation when possible saves 0.5 function evaluations over using only the secant method.

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 [268, Chap. 4.7].

Review Questions

6.2.1 Sketch a function $f(x)$ with a root in (a, b) such that regula falsi converges very slowly.

6.2.2 Outline how the secant method can be safeguarded by combining it with the bisection method.

6.2.3 What property should the function $f(x)$ have to be unimodal on the interval $[a, b]$?

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

Problems and Computer Exercises

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

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

6.2.2 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 in a modified step $\epsilon_{n+1} \approx -\epsilon_n$ when $\theta = 0.5$. Deduce that the resulting algorithm gives cubic convergence with three function evaluations and hence has efficiency index $E = 3^{1/3} = 1.4422\ldots$.[180]

6.2.3 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

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

6.2.4 The Muller–Traub 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 derived 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 [204, p. 159]). Hence

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

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

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

[180] 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 1950s.

6.2.5 The result in Problem 6.2.4 suggests that the Illinois method in Problem 6.2.2 should be 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

In Newton's method for solving an equation $f(x) = 0$ the curve $y = f(x)$ is approximated by its tangent at the point $(x_n, f(x_n))$, where x_n is the current approximation to the root. Assuming that $f'(x_n) \neq 0$, the next approximation x_{n+1} is then determined as the abscissa of the point of intersection of the tangent with the x-axis (see Figure 1.1.3).

When $f'(x)$ is available **Newton's method** is usually the method of choice. It is equivalent to replacing the equation $f(x) = 0$ by

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

which is obtained by taking the linear part of the Taylor expansion of $f(x)$ at x_n. Hence if $f'(x_n) \neq 0$, then

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

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

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 $f'(x) = x/2 - \cos x$. The following Newton iterates, starting from $x_0 = 1.8$, 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 doubles in each iteration until the limiting precision is reached. Although the initial approximation is not very good, already x_4 is correct to 12 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 $|Kh_n| \leq 1/2$, where K is an upper bound for $|f''/f'|$ in the neighborhood of the root. This restriction is seldom of practical importance. But rounding errors made in computing h_n must also be taken into account.

Note that when the root is approached, the *relative* precision in the computed values of $f(x_n)$ usually decreases. 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 $f'(x_n)$ for $n > 2$ without much effect on the accuracy. Such a simplification is of great importance when Newton's method is used on *systems* of nonlinear equations.

We now consider the **local** convergence of Newton's method, i.e., the convergence in a neighborhood of a simple root α.

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

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

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

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

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. □

The following classical theorem gives rigorous conditions for the quadratic convergence of Newton's method. In particular, *it is not necessary to assume the existence of a solution*.

Theorem 6.3.2 (*Ostrowski* [279, Theorem 7.1]).

Let $f(x)$ be a real function, $f(x_0)f'(x_0) \neq 0$, and put $h_0 = f(x_0)/f'(x_0)$. Assume that $f''(x)$ exists in $J_0 = \text{int}\,[x_0, x_0 + 2h_0]$, and that

$$2M|h_0| \leq |f'(x_0)|, \quad M = \sup_{x \in J_0} |f''(x)|. \tag{6.3.5}$$

Let the sequence x_k, $k = 1, 2, \ldots$, be generated by Newton's method:

$$x_{k+1} = x_k - f(x_k)/f'(x_k), \quad k = 0, 1, \ldots. \tag{6.3.6}$$

Then $x_k \in J_0$, and we have $\lim_{k\to\infty} x_k = \alpha$, where α is the only zero of $f(x)$ in J_0. Unless $\alpha = x_0 + 2h_0$, α is a simple zero.[181] *Further, it holds that*

$$|x_{k+1} - x_k| \le \frac{M}{2|f'(x_k)|}|x_k - x_{k-1}|^2, \quad k = 1, 2, \ldots, \tag{6.3.7}$$

$$|\alpha - x_{k+1}| \le \frac{M}{2|f'(x_k)|}|x_k - x_{k-1}|^2, \quad k = 1, 2, \ldots. \tag{6.3.8}$$

Proof. We have by (6.3.5) that $|f'(x) - f'(x_0)| \le |x - x_0|M$, and

$$|f'(x_1) - f'(x_0)| \le |h_0|M \le \frac{1}{2}|f'(x_0)|.$$

It follows that

$$|f'(x_1)| \ge |f'(x_0)| - |f'(x_1) - f'(x_0)| \ge \tfrac{1}{2}|f'(x_0)|. \tag{6.3.9}$$

Integrating by parts we have

$$\int_{x_0}^{x_1} (x_1 - x) f''(x)\, dx = -(x_1 - x_0) f'(x_0) + f(x_1) - f(x_0) = f(x_1).$$

Introducing a new variable by $x = x_0 + th$, this becomes

$$f(x_1) = h_0^2 \int_0^1 (1 - t) f''(x_0 + th)\, dt,$$

and it now follows that

$$|f(x_1)| = |h_0|^2 \int_0^1 (1 - t)|f''(x_0 + th)|\, dt \le M|h_0|^2 \int_0^1 (1 - t)\, dt = \tfrac{1}{2}M|h_0|^2. \tag{6.3.10}$$

Using (6.3.9) and (6.3.10) gives

$$|h_1| \le \frac{M|h_0|^2}{|f'(x_0)|}, \tag{6.3.11}$$

and applying (6.3.10) gives

$$\frac{2M|h_1|}{|f'(x_1)|} \le \frac{2(M|h_0|)^2}{|f'(x_0)||f'(x_1)|} \le \left(\frac{(2M|h_0|)}{|f'(x_0)|}\right)^2.$$

By (6.3.5) the expression in parenthesis is ≤ 1 and hence

$$2M|h_1| \le |f'(x_1)|, \quad |h_1| \le \tfrac{1}{2}|h_0|. \tag{6.3.12}$$

[181] It can be proved that if $\alpha = x_0 + 2h_0$, then $f(x)$ is a quadratic polynomial with the double root α; see [279, Chapter 40].

This shows that the point x_2 will not get beyond the distance $\frac{1}{2}|h_0|$ from x_1 and will remain in J_0, and int $[x_1, x_1 + 2h_1] \in J_0$.

We can now use induction to prove that the intervals $J_k = \text{int}\,[x_k, x_k + 2h_k]$, $k = 1, 2, \ldots$, lie in J_0, that $J_{k+1} \in J_k$, and that the length of J_{k+1} is at most half of the length of J_k. Since J_0 is closed it follows that

$$\lim_{k\to\infty} x_k = \alpha \in J_0.$$

To show that α is a root of $f(x)$ we note that from (6.3.6) it follows that $f'(x_k)(x_k - x_{k+1}) = f(x_k)$. Taking the limit it follows that $\lim_{k\to\infty} f(x_k) = f(\alpha) = 0$.

By (6.3.5) we have for all $x \in J_0$

$$|f'(x) - f'(x_0)| \le |x - x_0|M \le 2M|h_0|.$$

Assume that $|x - x_0| < |h_0|$, i.e., x_0 lies *in the interior* of J_0. Then

$$|f'(x) - f'(x_0)| < 2M|h_0| \le |f'(x_0)|,$$

so that $f'(x) \neq 0$ for all x in the interior of J_0. Therefore, α must be a simple root if it lies in the interior of J_0. Further, since $f'(x)$ does not vanish in J_0, $f(x)$ is strictly monotonic in J_0 and thus has only one root.

The assertion (6.3.8) is equivalent to

$$|h_k| \le \frac{M|h_{k-1}|^2}{2|f'(x_k)|}.$$

Since our starting assumption (6.3.5) is true for all $k \ge 0$, we have $|f(x_k)| \le \frac{1}{2}M|h_{k-1}|^2$, and hence the assertion follows using (6.3.6). To prove (6.3.8) we notice that α lies in an interval with center x_{k+1} and radius $|h_k|$, and use (6.3.8). □

Note that the relation (6.3.8) between the errors holds only as long as the roundoff errors in the calculations can be ignored. As pointed out in Sec. 6.1.3, the accuracy which can be achieved in calculating the root is always limited by the accuracy in 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))

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

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

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

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 rates of convergence for simple roots.) For $q > 2$ this is much slower even than for the bisection method! 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.

Note also that when $x_n \to \alpha$, $f(x_n) \to 0$ and $f'(x_n) \to 0$. Therefore, rounding errors may seriously affect the Newton correction when evaluated close to α, This clearly is related to the lower limiting accuracy for multiple roots as given by (6.1.10).

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

$$x_{n+1} = x_n - q\frac{f(x_n)}{f'(x_n)} \tag{6.3.13}$$

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 if the equation $f(x) = 0$ has a multiple root at $x = \alpha$, then α is *a simple root* of the equation

$$u(x) = 0, \quad u(x) = f(x)/f'(x). \tag{6.3.14}$$

Hence, if Newton's method is applied to this equation it will have a quadratic rate of convergence independent of the multiplicity of α as a root to $f(x) = 0$. The Newton iteration for $u(x) = 0$ becomes

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n) - f(x_n)f''(x_n)/f'(x_n)}, \tag{6.3.15}$$

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

Under certain regularity assumptions Newton's method is **locally convergent** from a sufficiently good initial approximation. But in general Newton's method is not **globally convergent**, i.e., it does not converge from an arbitrary starting point. This is not surprising since it is based on a series of local approximations.

It is easy to construct examples where Newton's method converges very slowly or not at all.

Example 6.3.2.

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 $x_0 = z$ such that $\tan z = 2z$, 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 $|x_0| < z$. The critical value can be shown to be $x_0 = 1.16556\ldots$.

Example 6.3.3.

In Example 5.2.5 we encountered the function $u(x)$ defined for $x \geq 0$ by

$$x = u \ln u. \tag{6.3.16}$$

The function $u(x)$ is related to Lambert's W-function $w(x)$ (see Problem 3.1.12 (d) and [370]), which is the inverse of the function $x(w) = we^w$. Clearly $u(x) = e^{w(x)}$. It can be shown that $u(x)$ is a smooth and slowly varying function x.

In Figure 6.3.1 we show a plot of the function $x = u \ln u$. For the inverse function $u(x)$ we have $u^p r(x) = 1/(1 + lnu(x))$. Clearly $u(0) = 1$, $u'(0) = 1$, and for $x > 0$ $u'(x)$, is positive and decreasing, while $u''(x)$ is negative and decreasing in absolute value.

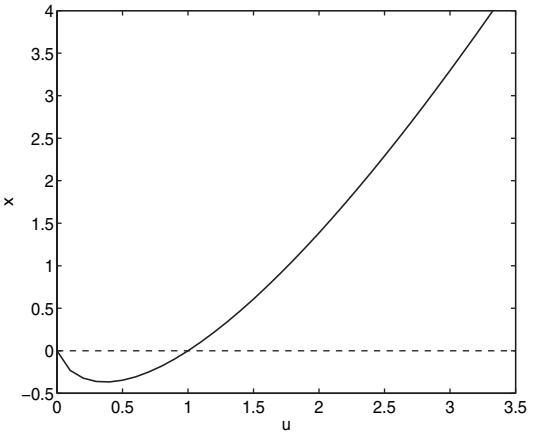

Figure 6.3.1. *The function $x = u \ln u$.*

To compute $u(x)$ to high accuracy for a given value of $x > 0$, we can use Newton's method to solve the equation

$$f(u) = u \ln u - x = 0.$$

This takes the form

$$u_{k+1} = \frac{u_k + x}{1 + \ln u_k}, \quad k = 0, 1, 2, \ldots.$$

For $x > 10$, we use as starting value u_0 the asymptotic approximation $u(x) \sim x/\ln x = u_0$, $x \to \infty$. For $0 < x \leq 10$ the approximation

$$u_0 = 1.0125 + 0.8577x - 0.129013x^2 + 0.208645x^3 - 0.00176148x^4 + 0.000057941x^5,$$

derived by Gautschi from a truncated Chebyshev expansion of $u(x)$, has a maximum error of about 1%. It is left to Problem 6.3.6 to investigate how efficient the outlined Newton method is for computing $u(x)$.

Global Convergence

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 6.3.3.

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 6.3.2. □

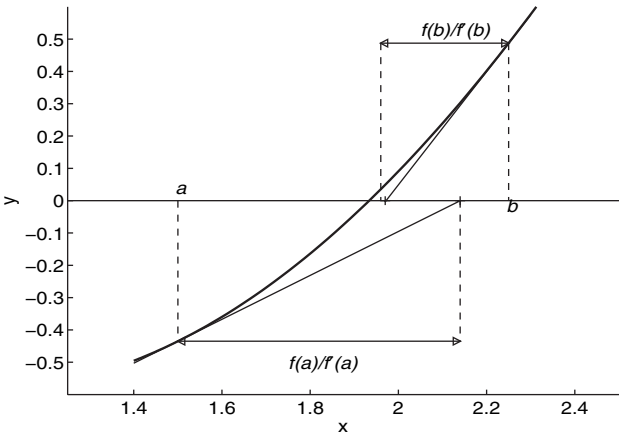

Figure 6.3.2. *A situation where Newton's method converges from any $x_0 \in [a, b]$.*

Lemma 6.3.4.

Let $[a, b]$ be an interval such that $f(a)f(b) < 0$. Assume that the so-called **Fourier conditions** *are satisfied, i.e.,*

$$f'(x)f''(x) \neq 0, \quad x \in [a, b], \tag{6.3.17}$$

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 6.3.2, 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. □

Newton's method can be safeguarded by taking a bisection step whenever a Newton step "fails" in some sense. Assume that initially $a < b$, $f(a)f(b) < 0$, and x is 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 $f(a')f(b') \le 0$.

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

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

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

$$(x - b)f'(x) < f(x) < (x - a)f'(x).$$

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

$$(x - a)f'(x) < f(x) < (x - b)f'(x).$$

In either case only one of the inequalities will be nontrivial depending on whether $f(x) > 0$.

6.3.2 Newton's 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 (see Sec. 3.1.2). Thus 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.

The geometry of the complex Newton iteration has been studied by Yao and Ben-Israel [387]. 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

$$\operatorname{grad} \phi = (\phi_x, \phi_y) = \frac{1}{\phi}\left(uu_x + vv_x, uu_y + vv_y\right), \tag{6.3.18}$$

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 [196, Sec. 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 (6.3.18) shows that the Newton step

$$z_{k+1} = z_k - f(z_k)/f'(z_k), \quad k = 0, 1, \ldots, \tag{6.3.19}$$

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

Theorem 6.3.5.

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 (6.3.19). *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 (x, y)-plane, then (x_{k+1}, y_{k+1}) is the point on L_k closest to (x_k, y_k).*

Proof. See [387]. □

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.

It is straightforward to generalize the results in Theorem 6.3.2 to the complex case. In the case of a complex function $f(x)$ of a complex variable the interval $I_0 = \text{int}\,[x_0, x_0+2h_0]$ can be replaced by a disk $K_0 : |z - z_1| \leq |h_0|$.

Theorem 6.3.6.

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

$$2\,|h_0|M_2 \leq |f'(z_0)|, \quad M_2 = \max_{z\in K_0} |f''(x)|. \tag{6.3.20}$$

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

$$|\zeta - z_{k+1}| \leq \frac{M_2}{2|f'(z_k)|}|z_k - z_{k-1}|^2, \quad k = 1, 2, \ldots. \tag{6.3.21}$$

A generalization of this theorem to systems of nonlinear equations is the famous Newton–Kantorovich theorem; see Volume II, Sec. 11.1.

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 (6.3.20) is satisfied will be described in Sec. 6.5.5.

6.3.3 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 [271].

Suppose that the function $f(x)$ is continuously differentiable. Using the notation in Sec. 2.5.3, let $f'([x_0])$ denote an interval containing $f'(x)$ for all x in a finite interval $[x_0] := [a, b]$. Define the Newton operator on $N[x]$ by

$$N([x]) := m - \frac{f(m)}{f'([x])}, \tag{6.3.22}$$

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

Theorem 6.3.7.

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

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

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

$$(m - f(m)/f'(\xi_1)) - a = -f(a)/f'(\xi_1),$$
$$b - (m - f(m)/f'(\xi_2)) = f(b)/f'(\xi_2).$$

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 $k = 0, 1, 2, \ldots$ a sequence of intervals $[x_{k+1}]$ given by

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

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 with the standard Newton method.

Example 6.3.4.

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

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

we obtain the sequence of intervals

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

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

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

$$0.5, 0.03125, 0.00036, \ldots.$$

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 6.3.7 holds. If this is the case, we continue the interval Newton iterations, and if we are close enough the iterations converge toward 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 processed. Intervals may be added or removed from the queue. When the queue is empty we are done.

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

6.3.4 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 example, when higher-order derivatives are easily computed. 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 derive 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

$$f(x_n) + hf'(x_n) + \frac{h^2}{2} f''(x_n) = 0, \quad h = x - x_n. \tag{6.3.23}$$

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

$$x_{n+1} = x_n - u(x_n) \cdot \frac{2}{1 + \sqrt{1 - 2t(x_n)}}, \tag{6.3.24}$$

where we have introduced the notations

$$u(x) = \frac{f(x)}{f'(x)}, \qquad t(x) = \frac{f(x_n)f''(x_n)}{(f'(x_n))^2} = u(x)\frac{f''(x)}{f'(x)}. \tag{6.3.25}$$

The iteration (6.3.24) is **Euler's iteration method**.

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

$$\frac{2}{1 + \sqrt{1 - 2t(x_n)}} \approx 1 + \frac{1}{2}t(x_n), \tag{6.3.26}$$

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

$$x_{n+1} = x_n - u(x_n)\left(1 + \frac{1}{2}t(x_n)\right). \tag{6.3.27}$$

A different method of cubic convergence is obtained by using a rational approximation of (6.3.26):

$$x_{n+1} = x_n - \cdot\frac{u(x_n)}{1 - \frac{1}{2}t(x_n)}. \tag{6.3.28}$$

This is **Halley's**[182] iteration method [179], which according to Traub [354] has the distinction of being the most frequently rediscovered iteration method. Halley's method has a simple geometric interpretation. Consider a hyperbola

$$h(x) = b + a/(x - c),$$

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

The methods (6.3.27) and (6.3.28) 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 [129] we consider an iteration function of the general form

$$\phi(x) = x - u(x)H(t(x)), \tag{6.3.29}$$

where $u(x)$ and $t(x)$ are defined by (6.3.25). Differentiating (6.3.29) and using $u'(x) = 1 - t(x)$ we get

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

[182] Edmond Halley (1656–1742), English astronomer, who predicted the periodic reappearance (ca 75 years) of a comet, which is now named after him.

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 (6.3.29) 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 (6.3.29) yields convergence of at least third order if

$$H(0) = 1, \qquad H'(0) = 1/2. \tag{6.3.30}$$

For Euler's and Halley's method we have

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

respectively, and both these methods satisfy (6.3.30). (Verify this!)

It is not very difficult to construct iteration methods of arbitrarily high order for solving $f(x) = 0$. It can be shown [354, 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:

$$0 = f(x_n + h) = f(x_n) + hf'(x_n) + \sum_{k=2}^{p-1} \frac{h^k}{k!} f^{(k)}(x_n) + O(h^p). \tag{6.3.31}$$

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 set $u = f(x_n)/f'(x_n)$, and write

$$u = -h - \sum_{k=2}^{p-1} a_k h^k, \quad a_k = \frac{f^{(k)}(x_n)}{k!\,f'(x_n)}, \quad k = 2 : p-1. \tag{6.3.32}$$

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:

$$h = -u - \sum_{k=2}^{p-1} c_k u^k + \cdots, \tag{6.3.33}$$

where the first few coefficients are

$$\begin{aligned} &c_2 = a_2, \qquad c_3 = 2a_2^2 - a_3, \qquad c_4 = 5a_2^3 - 5a_2a_3 + a_4, \\ &c_5 = 14a_2^4 - 21a_2^2a_3 + 6a_2a_4 + 3a_3^2 - a_5, \ldots. \end{aligned} \tag{6.3.34}$$

More coefficients can easily be determined; see Problem 3.1.12. This leads to the family of **Schröder methods** [316]. If f is analytic it can be shown that the method

$$x_{n+1} = x_n - u(x_n) - \sum_{k=2}^{p-1} c_k(x_n)u^k(x_n), \quad p \geq 2, \tag{6.3.35}$$

yields convergence order p for simple roots (see Henrici [196, p. 529]). Setting $p = 2$ gives Newton's method and for $p = 3$ we get the third-order method of Euler (6.3.27).

The Schröder methods make use of *polynomials* in u. As for the case $p = 3$, there are methods which instead use *rational expressions* in u. These can be derived from Padé approximants of the Schröder polynomials; see Traub [354, Sec. 5.2]. There are indications that methods which use rational approximations with about equal degree of numerator and denominator are best. For example, for $p = 4$, the rational approximation

$$1 + c_2u + c_3u^2 = \frac{c_2 + (c_2^2 - c_3)u}{c_2 - c_3u} + \mathcal{O}(u^3)$$

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

Example 6.3.5.

A kth-order iteration method (k odd) for the square root of c is

$$x_{n+1} = x_n \frac{x_n^{k-1} + \binom{k}{2}x_n^{k-3}c + \cdots + kc^{(n-1)/2}}{\binom{k}{1}x_n^{k-1} + \binom{k}{3}x_n^{k-3}c + \cdots + c^{(n-1)/2}}, \tag{6.3.36}$$

where the sums end in a natural way with zero binomial coefficients. For $k = 5$ we obtain the iteration

$$x_{n+1} = \frac{x_n^2 + 10c + 5(c/x_n)^2}{5x_n^2 + 10c + (c/x_n)^2}x_n, \quad n = 0, 1, 2, \ldots, \tag{6.3.37}$$

with order of convergence equal to five. For $k = 3$ we obtain Halley's method

$$x_{n+1} = \frac{x_n + 3c/x_n}{3x_n + c/x_n}x_n, \quad n = 0, 1, 2, \ldots, \tag{6.3.38}$$

which has cubic rate of convergence.

Using Halley's method with $c = 3/4$ and the initial approximation $x_0 = (\sqrt{2} + 2)/4$ (obtained by linear interpolation at $1/2$ and 1) we obtain the following result (correct digits in boldface):

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

Already two iterations give a result correct to 14 digits. Compared to Newton's method (see Example 1.1.1) we have gained one iteration. Since each iteration is more costly there is no improvement in efficiency.

We now introduce a rational family of iteration methods of arbitrary order, which is very convenient to use. First note that alternative derivation of Halley's method is as follows. Starting from (6.3.23) 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 (6.3.28). 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 functions of arbitrary order which we now present. Set $D_p(x) = \det(F_p)$, where

$$F_p(x) = \begin{pmatrix} f'(x) & \frac{f''(x)}{2!} & \dots & \frac{f^{(p-1)}(x)}{(p-1)!} & \frac{f^{(p)}(x)}{(p)!} \\ f(x) & f'(x) & \ddots & \ddots & \frac{f^{(p-1)}(x)}{(p-1)!} \\ 0 & f(x) & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \frac{f''(x)}{2!} \\ 0 & 0 & \dots & f(x) & f'(x) \end{pmatrix} \in \mathbf{R}^{p\times p} \tag{6.3.39}$$

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

$$x_{n+1} = x_n + B_p(x_n), \quad B_p(x) = -f(x)\frac{\det(F_{p-2}(x))}{\det(F_{p-1}(x))} \tag{6.3.40}$$

can be shown to be of order p for simple roots. Notice that for $f(x) = x^2 - c$ the matrix F_p becomes tridiagonal, which gives a simple iteration method of arbitrarily high order for the square root.

The determinant formula (6.3.40) is attractive since it leads to a simple implementation. Use Gaussian elimination without pivoting to compute the LU factorization $F_p(x_n) = L_pU_p$, where

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

Since L_p is unit lower triangular and U_p upper triangular (see Sec. 1.3.2) we have $\det(F_p) = u_{11}u_{22}\dots u_{pp}$. Hence the ratio

$$\det(F_p(x))/\det(F_{p-1}(x))$$

simply equals the *last diagonal element* u_{pp} in U_p. It follows that

$$x_{n+1}^{(k)} = x_n - f(x_n)/u_{kk}(x_n), \quad k = 1:p, \tag{6.3.41}$$

gives the result of one iteration step using a sequence of iteration formulas of order $k = 2:p+1$.

Note that the Gaussian elimination is simplified by the fact that $F_p(x_n)$ is a Hessenberg matrix. 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 6.3.16.

Methods using high-order derivatives are useful, especially when seeking zeros of a function satisfying a differential equation (usually of second order). Then higher-order derivatives can be calculated by differentiating the differential equation.

Example 6.3.6.

The Bessel function 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

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

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 $p = 5$ we compute

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

and form the Toeplitz matrix F_4. Computing the diagonal elements of U in its LU factorization, we obtain using (6.3.41) the following sequence of approximations to ξ:

$$\mathbf{2.40482}\,07503, \quad \mathbf{2.40482\,55}406, \quad \mathbf{2.40482\,55576}\,7553, \quad \mathbf{2.40482\,55576\,9573}.$$

Correct digits are shown in boldface. Hence the fifth-order method gives close to full IEEE double precision accuracy!

Review Questions

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

(b) Construct an example where Newton's method diverges, even though the equation has real roots.

6.3.2 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 $1/c$.

6.3.3 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 $k \geq 0$!

6.3.4 (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

6.3.1 (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.

6.3.2 The equation $x^3 - 2x - 5 = 0$ is of historical interest because it was the one used by Wallis[183] to present Newton's method to the French Academy. Determine the roots of this equation.

Hint: It has one real and two complex roots.

6.3.3 Use Newton's method to determine the positive root of the equation to six correct decimals: (a) $x = 1 - e^{-2x}$; (b) $x \ln x - 1 = 0$.

6.3.4 Determine the unique positive real root of the equation $x^q - x - 1 = 0$ for $q = 2 : 8$.

6.3.5 (a) Consider the Newton iteration used in Example 6.3.5 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 $\lim_{n\to\infty} = \sqrt{c}$ (compare Figure 1.1.2).

(b) In Example 6.3.5 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 $[1/2, 1]$ using this initial approximation. Show that the influence of rounding errors is negligible.

6.3.6 (a) Investigate how many Newton iterations are required to compute the inverse function $u(x)$ of $x = u \ln u$, to 12 digits of accuracy in the interval $0 < x \leq 20$. Use the starting approximations given in Example 6.3.3.

(b) Lambert's W-function equals $w(x) = \ln u(x)$, where $u(x)$ is the inverse function in (a). Compute and plot the function $w(x)$ for $0 < x \leq 20$.

6.3.7 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.

6.3.8 (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.

6.3.9 The Cartesian coordinates of a planet in elliptic orbit at time t are equal to $ea(\sin(x), \cos(x))$, where a is the semimajor axis, and e the eccentricity of the ellipse. Using

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

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:

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

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

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

(a) Use bisection (cf. Algorithm 6.1) 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) Repeat (b), but perform one step of the modified Newton's method (6.3.13) 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)?

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

$$x_{n+1} = x_n - \frac{u(x_n)}{1 - t(x_n)}, \quad t(x_n) = \frac{f(x_n) f''(x_n)}{(f'(x_n))^2}. \tag{6.3.42}$$

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.

6.3.12 (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 are the efficiency indexes of Newton's and Halley's methods, 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 $f(x) = x^k - c = 0$, $c > 0$.

6.3.13 (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.

6.3.14 (Bailey et al. [11]) 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 $k = 1, 2, 3, \ldots$ compute

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

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?

6.3.15 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.

6.3.16 (a) Compute $\det(F_p(x))$ in (6.3.39) for $p = 3$ and write down the corresponding rational fourth-order iteration method in terms of u, a_2, a_3 in (6.3.34).

(b) Implement in MATLAB the iteration method (6.3.40) 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.

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

6.3.18 (a) Given $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_n > 0$, $c_1, c_2, \ldots, c_n$, and a parameter $\gamma > 0$, consider the equation

$$f(\lambda) = \left(\sum_{i=1}^{n} \frac{\sigma_i^2 c_i^2}{(\sigma_i^2 + \lambda)^2}\right)^{1/2} = \gamma. \tag{6.3.43}$$

Each term in the sum is a convex and strictly decreasing function of λ for $\lambda > 0$. Show that this also holds for $f(\lambda)$. Further, show that if $f(0) > \gamma$ (6.3.43) has a unique root $\lambda^* > 0$.

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

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

Show that with $\lambda_0 = 0$, this gives a strictly increasing sequence $\{\lambda_k\}_{k=0}^{\infty}$ converging to the solution. Derive an explicit expression for h_k.

(c) The Newton method suggested in (b) will converge slowly when $\lambda_k \ll \lambda^*$. A more efficient method is obtained by instead applying Newton's method to the equation $h(\lambda) = 1/f(\lambda) = 1/\gamma$. Show that this iteration can be written

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

where h_k is the Newton step in (b).

(d) Let

$$\sigma_i = 1/i^2, \qquad c_i = 1/i^2 + 0.001, \quad i = 1:20.$$

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 with $p = 1$ and $p = -1$.

6.4 Finding a Minimum of a Function

6.4.1 Introduction

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

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

which is closely related to that of solving a scalar equation.

Probably the most common use of unidimensional minimization is in "line search" procedures in multidimensional minimization of a function $\phi(z)$ of n variables, $z \in \mathbf{R}^n$. 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 step length λ is to be determined. In this application, however, the minimization is rarely performed accurately.

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 minimum** (i.e., the 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. However, this approach is neither efficient or safe.

Suppose we want to find a local minimum of $g(x)$ in a given interval $[a, b]$. 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 finding the zeros of $g'(x)$ in I. However, since g' also vanishes at a point of maximum and inflection, it is necessary to check, for example, the second derivative, to see if the point found really is a minimum.

6.4.2 Unimodal Functions and Golden Section Search

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 $[-1, 1]$.

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 $1/2$. But $d - c = (1 - 2t)(b - a)$ must not be too small for the available precision in evaluating $g(x)$.

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 first case. Then t must satisfy the following relation (cf. above and Figure 6.4.1):

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

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 $(0.618034\ldots)^{20} \approx 0.661 \cdot 10^{-5}$.

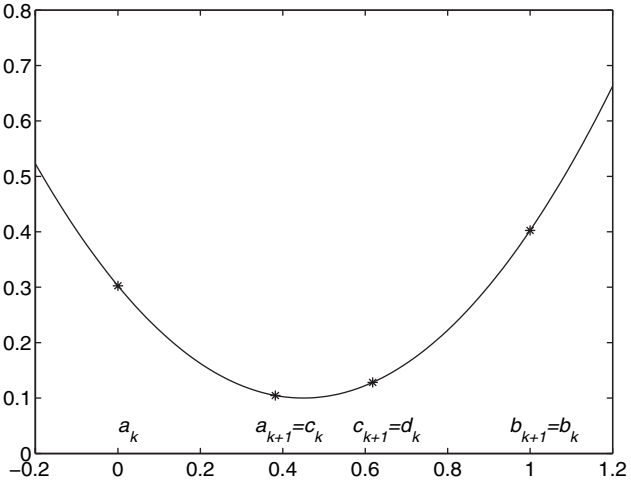

Figure 6.4.1. *One step of interval reduction, $g(c_k) \geq g(d_k)$.*

ALGORITHM 6.2. *Golden Section Search.*

The function `goldsec` computes an approximation $m \in I$ to a local minimum of a given function in an interval $I = [a, b]$, with an error less than a specified tolerance τ.

```
function xmin = goldsec(fname,a,b,delta);
%
t = 2/(3 + sqrt(5));
c = a + t*(b - a);
d = b - t*(b - a);
fc = feval(fname,c);
fd = feval(fname,d);
while (d - c) > delta*max(abs(c),abs(d))
   if fc >= fd    % Keep right endpoint b
      a = c;
      c = d;
      fc = fd;
      d = b - t*(b - a);
      fd = feval(fname, d);
    else          % Keep left endpoint a
      b = d;
      d = c;
      fd = fc;
      c = a + t*(b - a);
      fc = feval(fname, 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. Therefore, we assume that in Definition 6.4.1 the points c and d satisfy $|c - d| >$ tol, where

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

is chosen as a combination of absolute and relative tolerance. Then 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

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

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 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 Sec. 6.2.4.

Since linear functions generally have no minimum the simplest choice is to use a second degree polynomial (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. (6.2.12))

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

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 gives

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

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 $u = a$, $w = b$; if $g(c) < g(d)$, then $v = c$, otherwise $v = d$, where c and d are the two golden section points.

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

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

Consider parabolic interpolation at the points x_{i-2}, x_{i-1}, x_i, $i = 2, 3, \ldots$, 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 $p = 1.3247\ldots$ 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:

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

where e is the value of the second last cycle. 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 [44, Chapter 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 positions 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.

Review Questions

6.4.1 How many steps are needed in golden section search to reduce an initial interval $[a, b]$ by a factor of 10^{-6}?

6.4.2 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?

6.4.3 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

6.4.1 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.

6.4.2 Modify the algorithm `goldsec` to use parabolic interpolation instead of golden section if this gives a point within the interval.

6.4.3 (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 6.4.2 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).

6.4.4 (Brent [44, Sec. 5.6]) The function

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

has poles at $x = 1^2, 2^2, \ldots, 20^2$. 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 Algebraic Equations

6.5.1 Some Elementary Results

The problem of solving an algebraic 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 equation 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 sixteenth 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 algebraic equations of degree $n > 4$. But even the existing formulas for $n \leq 4$ are not, in general, suitable for numerical evaluation of the roots. In Sec. 2.3.4, it was shown that care must be taken to avoid cancellation even in the case of a quadratic equation.

Despite the absence of closed solution formulas, the **fundamental theorem of algebra** states that every algebraic equation has at least one root. More precisely, if a polynomial vanishes nowhere in the complex plane, then it is identically constant. The **remainder theorem** states that when a polynomial $p(z)$ is divided by $z - r$ the remainder is $p(r)$, i.e.,

$$p(z) = (z-r)p_1(z) + p(r). \tag{6.5.2}$$

It follows that r_1 is a zero of $p(z)$ if and only if $z - r_1$ divides $p(z)$. If $p(z)$ is of degree n and $p(r_1) = 0$, then $p_1(z)$ in (6.5.2) is of degree $n-1$. But if $n > 1$, then $p_1(z)$ must vanish for some r_2, and hence has a linear factor $z - r_2$. Continuing, we find that an algebraic equation $p(z) = 0$ of degree n has exactly n roots, counting multiplicities, and it holds that[184]

$$p(z) = a_0(z-r_1)(z-r_2)\cdots(z-r_n). \tag{6.5.3}$$

By this representation it also follows that if the coefficients $a_0, a_1, \ldots, a_n$ are real, then eventual complex roots must occur in conjugate pairs. Hence, if $p(z)$ has the complex zero $s + it$ it also has the zero $s - it$ and therefore contains the quadratic factor $(z-s)^2 + t^2$ which is positive for all real values of z.

[184]Note the corollary that if a polynomial $p(z)$ of degree at most n vanishes in $n+1$ distinct points, it must be identically zero, a result used repeatedly in Chapters 3 and 4.

Solving algebraic 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. Algebraic equations of high degree ($n > 100$) occur in computer algebra. Such problems usually require symbolic, or high multiple-precision computations. Algorithms for such problems are still a subject of research.

Let $p(z)$ be the polynomial (6.5.3) with roots $r_1, r_2, \ldots, r_n$, not necessarily equal. Comparing with the coefficients of z^{n-k} in the representations (6.5.1) we find that $(-1)^k a_k/a_0$ is the sum of the products of the roots r_i taken k at a time. Thus we obtain the following relations between the coefficients and zeros of a polynomial:

$$\sum_i r_i = -\frac{a_1}{a_0}, \qquad \sum_{i<j} r_i r_j = \frac{a_2}{a_0}, \qquad \sum_{i<j<k} r_i r_j r_k = -\frac{a_3}{a_0}, \ldots,$$

$$r_1 r_2 \cdots r_n = (-1)^n \frac{a_n}{a_0}. \tag{6.5.4}$$

The functions on the left sides of (6.5.4) are called **elementary symmetric functions** of the variables $r_1, r_2, \ldots, r_n$, since interchanging any of the variables will not change the functions. A classical result says that any symmetric polynomial in the roots $r_1, r_2, \ldots, r_n$ can be expressed as a polynomial in the elementary symmetric functions in (6.5.4).

Other commonly used symmetric functions are the power sums

$$S_k = \sum_{i=1}^{n} r_i^k, \quad k = \pm 1, \pm 2, \ldots. \tag{6.5.5}$$

The **Newton formulas** give the connection between the coefficients of the polynomial $p(z)$ (with $a_0 = 1$), and the power sums:

$$\begin{aligned} &S_1 + a_1 = 0, \\ &S_2 + a_1 S_1 + 2a_1 = 0, \\ &S_3 + a_1 S_2 + a_2 S_1 + 3a_3 = 0, \\ &\quad \vdots \\ &S_{n-1} + a_1 S_{n-2} + \cdots + a_{n-2} S_1 + (n-1)a_{n-1} = 0. \end{aligned} \tag{6.5.6}$$

For a proof see Householder [204, Sec. 1.3].

If $a_n \neq 0$, setting $z = 1/y$ in $p(z)$ gives the **reciprocal polynomial**

$$q(y) = y^n p(1/y) = a_n y^n + \cdots + a_1 y + a_0. \tag{6.5.7}$$

The zeros of the reciprocal polynomial are $1/r_1, 1/r_2, \ldots, 1/r_n$ and from (6.5.4), giving the Newton formulas,

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

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. Sec. 1.2.2. If we set

$$\begin{aligned} p(z) &= (z - w)q(z) + b_n, \\ q(z) &= b_0 z^{n-1} + b_1 z^{n-2} + \cdots + b_{n-1}, \end{aligned} \tag{6.5.8}$$

then the sequence $\{b_i\}_{i=0}^n$ satisfies the recursion

$$b_0 = a_0, \qquad b_i = b_{i-1}w + a_i, \quad i = 1 : n. \tag{6.5.9}$$

Here $p(w) = b_n$ is the remainder and $q(z)$ is the quotient polynomial when dividing $p(z)$ with $(z - w)$. Differentiating (6.5.8) we get

$$p'(z) = (z - w)q'(z) + q(z), \tag{6.5.10}$$

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

$$\begin{aligned} q(z) &= (z - w)r(z) + c_{n-1}, \\ r(z) &= c_0 z^{n-2} + c_1 z^{n-3} + \cdots + c_{n-2}. \end{aligned}$$

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 thus $\frac{1}{2}p''(w) = q'(w) = d_{n-2}$, 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 6.5.2 (b)).

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

$$p(w + s) = q(s) = c_0 s^n + c_1 s^{n-1} + \cdots + c_n. \tag{6.5.11}$$

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**,[185] states that the number of positive roots is either given by the number of variations in sign in the sequence $a_0, a_1, \ldots, a_n$, 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. 6.5.6 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. For these we refer to surveys in the literature.

6.5.2 Ill-Conditioned Algebraic Equations

Let $a = (a_1, a_2, \ldots, a_n)^T \in \mathbf{C}^n$ be the coefficient vector of a monic polynomial $p(z)$ ($a_0 = 1$) and denote the set of zeros of p by $Z(p) = \{z_1, z_2, \ldots, z_n\}$. In general, the best we can expect from any numerical zero-finding algorithm is that it computes the zeros of a nearby polynomial $\hat{p}(z)$ with slightly perturbed coefficients $\hat{a}$. Following Trefethen [358] we define the ϵ-pseudo zero set of $p(z)$ by

$$Z_\epsilon = \{z \in \mathbf{C} \mid z \in Z(\hat{p}) \text{ for some } \hat{p} \text{ with } \|\hat{a} - a\| \le \epsilon\}. \tag{6.5.12}$$

These sets describe the conditioning of the zero-finding problem. Depending on the chosen norm they correspond to a coefficientwise or normwise perturbation of the coefficient vector a. The shape of these sets is studied in [353].

The sensitivity of polynomial zeros to perturbations in the coefficients was illustrated in a famous example by Wilkinson in the early 1960s.[186] The paper [379] contains an extensive discussion of numerical problems in determining roots of polynomial equations. Wilkinson considered the polynomial

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

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

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

[186] Wilkinson received the Chauvenet Prize of the Mathematical Association of America 1987 for this exposition of the ill-conditioning of polynomial zeros.

from those of $p(z)$. In fact, correct to nine decimal places, the perturbed zeroes are the following.

$$\begin{array}{rl} 1.000000000 & 10.095266145 \pm 0.643500904i \\ 2.000000000 & \\ 3.000000000 & 11.793633881 \pm 1.652329728i \\ 4.000000000 & \\ 4.999999928 & 13.992358137 \pm 2.518830070i \\ 6.000006944 & \\ 6.999697234 & 16.730737466 \pm 2.812624894i \\ 8.007267603 & \\ 8.917250249 & 19.502439400 \pm 1.940330347i \\ 20.846908101 & \end{array}$$

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 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 Sec. 2.3.2. Hence by (6.1.7) the limiting 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 $\epsilon_\alpha = 1.89 \cdot 10^{16}$. But the changes in this example are so large that this linearized perturbation theory does not apply! For a more detailed discussion of the conditioning of algebraic equations in general and the Wilkinson polynomial, we refer to Gautschi [144, Sec. 4].

It should be emphasized that although a problem may be given in the form (6.5.1), it is often the case that the coefficients of $p(z)$ are not the original data. *Then it may be better to avoid computing them.* An important case is when the polynomial is the **characteristic polynomial** of a matrix $A \in \mathbf{R}^{n\times n}$:

$$p_A(z) = \det(zI - A) = \begin{vmatrix} z - a_{11} & -a_{12} & \cdots & -a_{1n} \\ -a_{21} & z - a_{22} & \cdots & -a_{2n} \\ \vdots & \ddots & \vdots & \vdots \\ -a_{n1} & -a_{n2} & \cdots & z - a_{nn} \end{vmatrix}. \tag{6.5.13}$$

The n roots of $p_A(z) = 0$ are the eigenvalues of A, and *the problem is an eigenvalue problem in disguise*. Then the original data are the elements of the matrix A and numerical values of $p_A(z)$ can then be evaluated much more accurately directly from (6.5.13). It is important to realize that, even when the roots (eigenvalues) are well determined by the elements of A

and well separated, they can be *extraordinarily sensitive to small relative perturbations in the coefficients of* $p_A(z)$.

In classical (i.e., before 1960) methods of linear algebra, the eigenvalues of a matrix are often found by first computing the coefficients of the characteristic polynomial. This is not in general a good idea and one of the highly developed modern eigenvalue algorithms should be used (consult Volume II, and references therein). Note also that if the coefficients of the characteristic polynomial $\det(zI - A)$ are required, these are often best computed by first computing the eigenvalues λ_i of A and then forming

$$p(z) = \prod_{i=1}^{n} (z - \lambda_i). \tag{6.5.14}$$

Example 6.5.1.

Another example where computing the coefficients of the polynomial should be avoided is the following. Suppose the largest positive root of the algebraic 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 one.

This is a poor result. However, by writing the equation in the form

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

and solving this by the iteration $x_0 = 1$, $x_{k+1} = \phi(x_k)$, one can get the root $\alpha = 1.053416973823$ to full accuracy.

Turning the tables, a polynomial zero-finding problem can be transformed into an eigenvalue problem. The **companion matrix** (or Frobenius matrix) to the polynomial $p(z)$ in (6.5.1), normalized so that $a_0 = 1$, is defined as

$$C = \begin{pmatrix} -a_1 & -a_2 & \cdots & -a_{n-1} & -a_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}. \tag{6.5.15}$$

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

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

Thus the roots of $p(z) = 0$ can be computed by applying an eigenvalue algorithm to C. This trick is used in several of the zero-finding algorithms in current use; see Sec. 6.5.5.

For example, 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 fast as competing algorithms and can be more accurate. Further problems with overflow or underflow are avoided.

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.

6.5.3 Three Classical Methods

In this section we describe three classical methods which are mainly of theoretical interest, but may be useful in special situations.

The idea in **Graeffe's method** is to replace equation (6.5.1) by an equation whose roots are the square of the roots of (6.5.1). By iterating this procedure roots which are of unequal magnitude become more and more separated and therefore, as we shall see, more easily determined.

If the monic polynomial $p_1(z) = z^n + a_1 z^{n-1} + \cdots + a_n$ has zeros z_i, $i = 1 : n$, then

$$p_1(z) = (z - z_1)(z - z_2) \cdots (z - z_n).$$

Setting $y = z^2$, it follows that

$$p_2(y) = (-1)^n p_1(z) p_1(-z) = (y - z_1^2)(y - z_2^2) \cdots (y - z_n^2);$$

i.e., $p_2(y)$ is a polynomial of degree n whose zeros equal the square of the zeros of $p_1(z)$. The coefficients of

$$p_2(y) = y^n + b_1 y^{n-1} + \cdots + b_n$$

are obtained by convolution of the coefficients of $p_1(z)$ and $p_1(-z)$; see Theorem 3.1.6. This gives

$$b_0 = a_0^2, \qquad (-1)^k b_k = a_k^2 + \sum_{j=1}^{k} (-1)^j 2 a_{k-j} a_{k+j}, \quad k = 1 : n. \tag{6.5.16}$$

Assume now that after repeated squaring we have obtained an equation

$$p_{m+1}(w) = w^n + c_1 w^{n-1} + \cdots + c_n,$$

with zeros $|\alpha_j| = |z_j|^{2^m}$, such that $|\alpha_1| \gg |\alpha_2| \gg \cdots \gg |\alpha_n|$. Then we can use the relations (6.5.4) between the coefficients and zeros of a polynomial to deduce that

$$c_1 = -\sum_i \alpha_i \approx -\alpha_1, \qquad c_2 = \sum_{i<j} \alpha_i \alpha_j \approx \alpha_1 \alpha_2, \qquad c_3 = \sum_{i<j<k} \alpha_i \alpha_j \alpha_k \approx \alpha_1 \alpha_2 \alpha_3, \ldots,$$

and therefore

$$|z_1| \approx (-c_1)^{1/2^m}, \qquad |z_2| \approx (-c_2/c_1)r^{1/2^m}, \qquad |z_3| \approx (-c_3/c_2)^{1/2^m}, \ldots.$$

Example 6.5.2.

Squaring the polynomial $p_1(z) = z^3 - 8z^2 + 17z - 10$ three times gives

$$p_4(w) = w^3 - 390{,}882w^2 + 100{,}390{,}881w - 10^8.$$

We have $2^m = 8$ and approximations to the roots are

$$\begin{aligned} |z_1| &\approx \sqrt[8]{390{,}882} = 5.00041, \\ |z_2| &\approx \sqrt[8]{100{,}390{,}881/390{,}882} = 2.00081, \\ |z_3| &\approx \sqrt[8]{10^8/100{,}390{,}881} = 0.999512. \end{aligned}$$

The exact roots are 5, 2, and 1.

In **Bernoulli's method** for obtaining the zeros of

$$p(z) \equiv z^n + a_1 z^{n-1} + \cdots + a_n = 0,$$

one considers the related difference equation

$$y_{n+k} + a_1 y_{n+k-1} + \cdots + a_n y_k = 0, \tag{6.5.17}$$

which has $p(z)$ as its characteristic equation. If $p(z)$ only has simple roots $z_1, \ldots, z_n$, then by Theorem 3.3.12 the general solution of (6.5.17) is given by

$$y_k = \sum_{j=1}^{n} c_j z_j^k,$$

where $c_1, c_2, \ldots, c_n$ are constants which depend on the initial conditions.

We assume in the following that the roots are ordered in magnitude so that

$$|z_1| > |z_2| \geq \cdots \geq |z_n|,$$

where the root of largest magnitude z_1 is distinct. We say that z_1 is a **dominant root**. Assuming that the initial conditions are chosen so that $c_1 \neq 0$, we have

$$y_k = c_1 z_1^k \left[1 + \sum_{j=2}^{n} \frac{c_j}{c_1} \left(\frac{z_j}{z_1} \right)^k \right] = c_1 z_1^k \left[1 + O\left(\frac{|z_2|}{|z_1|} \right)^k \right].$$

If z_1 is real it follows that

$$\lim_{n\to\infty} \frac{y_k}{y_{k-1}} = z_1 \left[1 + O\left(\frac{|z_2|}{|z_1|} \right) \right]. \tag{6.5.18}$$

Note that the rate of convergence is linear and depends on the ratio $|z_2/z_1|$. In practice measures to avoid overflow or underflow must be included. By initially applying a few steps of Graeffe's root-squaring method, convergence can be improved.

The relation (6.5.18) holds also if z_1 is a real multiple root. However, if the root of largest magnitude is complex the method needs to be modified. It is difficult to safeguard against all possible cases.

Bernoulli's method is closely related to the **power method** for computing approximate eigenvalues and eigenvectors of a matrix. In this application a powerful modification is to shift all the eigenvalues so that the desired eigenvalue is close to zero. If we then apply the power method to the inverse matrix rapid convergence is assured.

In **Laguerre's method**[187] 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

$$p(z_k) = r(z_k), \qquad p'(z_k) = r'(z_k), \qquad p''(z_k) = r''(z_k). \tag{6.5.19}$$

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 (6.5.19) 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!)

$$z_{k+1} = z_k - \frac{np(z_k)}{p'(z_k) \pm \sqrt{H(z_k)}}, \tag{6.5.20}$$

where

$$H(z_k) = (n-1)^2[p'(z_k)]^2 - n(n-1)p(z_k)p''(z_k).$$

[187] Edmond Nicolas Laguerre (1834–1886), French mathematician at École Polytechnique, Paris and best known for his work on orthogonal polynomials.

The sign in the denominator in (6.5.20) 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 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. But 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

$$z_1 = \frac{-na_n}{a_{n-1} \pm \sqrt{H(z_0)}}, \quad H(z_0) = (n-1)^2 a_{n-1}^2 - 2n(n-1)a_n a_{n-2}, \tag{6.5.21}$$

where the sign is to be chosen so 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 6.5.3.

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 (6.5.21) 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 $\pm i$,

$$z_2 = -0.00849761051 + 1.01435422762i,\ z_3 = -0.00011503062 + 1.00018804502i,$$
$$z_4 = -0.00000002143 + 1.00000003279i,\ z_5 = -0.00000000000 + 1.00000000000i.$$

6.5.4 Deflation and Simultaneous Determination of Roots

Suppose we have found a root α to the equation $p(z) = 0$. Then taking $z_k = \alpha$ in (6.5.9)–(6.5.8) 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 from converging 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 [379] shows that if the coefficients of the quotient polynomials are computed by the recursion (6.5.9), 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 $z_0 = 0$. But 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 [255]. 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 $\xi_1, \ldots, \xi_j$ to j roots of $p(z) = 0$. Then 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

$$z_{k+1} = z_k - \frac{p(z_k)}{p'(z_k) - \sum_{i=1}^{j} p(z_k)/(z_k - \xi_i)}, \tag{6.5.22}$$

which is the **Newton–Maehly method**. This iteration has the advantage that it is not sensitive to the accuracy in the approximations of the previous roots $\xi_1, \ldots, \xi_j$. Indeed, the iteration (6.5.22) is locally quadratically convergent to simple zeros of $p(z)$ for *arbitrary* values of $\xi_1, \ldots, \xi_j$.

For the removal of a linear factor by deflation it is necessary that the zero be computed to full working accuracy, since otherwise the remaining approximative zeros can be

meaningless. This is a disadvantage if only low accuracy is required. An alternative to deflation is to use an iterative method 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 of $p(z)$. A new set of approximations are then computed from

$$\xi_i^{(k+1)} = \xi_i^{(k)} - p(\xi_i^{(k)}) \Big/ \left[a_0 \prod_{\substack{j=1 \\ j\neq i}}^{n} (\xi_i^{(k)} - \xi_j^{(k)}) \right], \quad i = 1 : n. \tag{6.5.23}$$

This is **Weierstrass' method**, 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 [227].

It is possible to accelerate Weierstrass' method by using the new approximations of the roots in (6.5.23) as they become available. This leads to the iteration

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

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 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 (6.5.25) applied to the reciprocal polynomial.

The zeros z_i, $i = 1 : n$, of a polynomial of degree n,

$$p(z) = b_0 + b_1 z + \cdots + b_n z^n, \quad (b_0 b_n \neq 0), \tag{6.5.24}$$

are the poles of the rational function $r(z) = 1/p(z)$. Hence the progressive form of the qd algorithm, described in Sec. 3.5.5, can be used to simultaneously determine the zeros. This method can be considered as a modern, more powerful version of Bernoulli's method.

In the qd scheme for $r(z)$ the values in the first two rows can be expressed in terms of the coefficients of $p(z)$ as

$$q_1^{(0)} = -b_1/b_0, \qquad q_k^{(1-k)} = 0, \quad k > 1,$$

$$e_k^{(1-k)} = b_{k-1}/b_k, \quad k = 1 : n-1.$$

Further, since r has a zero of order n at infinity,

$$e_n^{(m)} = 0, \quad m \geq -n.$$

Thus the qd scheme is flanked on both sides by a column of zeros. This allows the extended qd scheme for $r(z)$ to be constructed row by row. It can be shown that a sufficient condition for the qd scheme to exist is that the zeros are positive and simple. Then the kth q-column tends to z_k^{-1}. By reversing the order of the coefficients, i.e., by considering the polynomial $z^k p(z^{-1})$ instead, we can also construct a scheme where the q-columns tends to the zeros z_m.

Example 6.5.4.

The Laguerre polynomial of degree four is

$$L_4(z) = \frac{1}{24}(24 - 96z + 72z^2 - 16z^3 + z^4).$$

The corresponding qd scheme is shown in Table 6.5.1.

Table 6.5.1. *The qd scheme for computing the zeros of $L_y(z)$.*

	16.00000000	0	0	0
0	−4.50000000	−1.33333333	−0.25000000	0
	11.50000000	3.16666667	1.08333333	0.25000000
0	−1.23913043	−0.45614035	−0.05769231	0
	10.26086957	3.94965675	1.48178138	0.30769231
0	−0.47697126	−0.17112886	−0.01197982	0
	9.78389831	4.25549915	1.64093042	0.31967213
0	−0.20745829	−0.06598769	−0.00233381	0
	9.57644002	4.39696974	1.70458430	0.32200594
0	−0.09525333	−0.02558161	−0.00044087	0
	9.48118669	4.46664146	1.72972504	0.32244681
	⋮	⋮	⋮	⋮
	9.39507749	4.53661379	1.74576103	0.32254769
0	−0.00000340	−0.00000004	0.00000000	0
	9.39507409	**4.53661**715	**1.74576108**	**0.32254769**

The zeros of $L_4(z)$ are correctly rounded to eight decimals:

$$9.39507091, \quad 4.53662030, \quad 1.74576110, \quad 0.32254769.$$

Note that the convergence of $e_i^{(n)}$ is linear with rate (z_{i+1}/z_i), $i = 1 : 3$, and can be accelerated with Aitken extrapolation.

6.5.5 A Modified Newton Method

There are three competing methods in current use for determining all zeros of a given polynomial. The Jenkins–Traub method [211], used in the IMSL library, is equivalent to a so-called variable-shift Rayleigh quotient iteration for finding the eigenvalues and eigenvectors of the companion matrix. By taking advantage of the matrix structure the work per iteration can be reduced to $O(n)$. A three stage procedure is used, each stage being characterized by the type of shift used. The code CPOLY (see [212]) is available via Netlib; see Sec. C.7 in Online Appendix C. For a description of the Jenkins–Traub method we refer to Ralston and Rabinowitz [296, Sec. 8.11].

The MATLAB zero-finding code `roots` applies the QR algorithm, which is a standard method for solving eigenvalue problems, to a balanced companion matrix. The balancing involves a diagonal similarity transformation,

$$\tilde{A} = DAD^{-1}, \quad D = \text{diag}\,(d_1, d_2, \ldots, d_n),$$

which preserves the eigenvalues. The aim is to reduce the norm of A and thereby reduce the condition number of its eigenvalue problem. The balancing algorithm used is that of Parlett and Reinsch [286].

Another excellent algorithm is the modified Newton algorithm PA16, due to Madsen and Reid [253, 254], used by NAG Library. In its first stage it uses the Newton formula to find a search direction for minimizing $|p(z)|$. Once the iterates are close to a zero it enters stage 2 and switches to standard Newton. This will be described in more detail below.

Theoretical and experimental comparisons of the three algorithms above are given in [353]. It is shown that the Jenkins–Traub, Madsen–Reid, and QR algorithms all have roughly the same stability properties. The highest accuracy is typically achieved by PA16 and the next best by `roots`. A possible drawback with the QR algorithm is that it requires $\mathcal{O}(n^3)$ work and $\mathcal{O}(n^2)$ memory. Both the Madsen–Reid and Jenkins–Traub require only $\mathcal{O}(n^2)$ work $\mathcal{O}(n)$ memory. Since one is rarely interested in solving polynomial equations of high degree this is usually not important.

We now describe the modified Newton method due to Madsen [253]. By including a one-dimensional search along the Newton direction this method achieves good global convergence properties and is also effective 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

$$z_1 = \frac{1}{2\rho}\frac{\delta z_0}{|\delta z_0|}, \qquad \rho = \max_{1\le k\le n}\left(\frac{|a_{n-k}|}{|a_n|}\right)^{1/k}. \tag{6.5.25}$$

This assures that $|z_1|$ is less than the modulus of any zero of $p(z)$ (see [204, 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

$$h_k = -p(z_k)/p'(z_k) \tag{6.5.26}$$

is computed. The next tentative step is taken as

$$\delta z_k = \begin{cases} h_k & \text{if } |h_k| \leq 3|z_k - z_{k-1}|, \\ 3|z_k - z_{k-1}|e^{i\theta} h_k/|h_k| & \text{otherwise,} \end{cases}$$

where 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 ($z_k = z_{k-1}$) the search direction is changed and the step size reduced. In this case the tentative step is chosen to be

$$\delta z_k = -\frac{1}{2} e^{i\theta} \delta z_{k-1}.$$

Repeated use of this is sure to yield a good search direction.

3. Once the tentative step δz_k has been found the inequality $|p(z_k + \delta z_k)| < |p(z_k)|$ is tested. If this is satisfied the numbers

$$|p(z_k + p\,\delta z_k)|, \quad p = 1, 2, \ldots, n,$$

are calculated as long as these are strictly decreasing. Note that if we are close to a multiple root of multiplicity m 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 6.3.3 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. (6.3.20). This inequality can be approximated using already computed quantities by

$$2\,|f(z_k)||f'(z_k) - f'(z_{k-1})| \leq |f'(z_k)|^2 |z_{k-1} - z_k|. \tag{6.5.27}$$

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 [253, 254].

6.5.6 Sturm Sequences

Precise information about the number of real zeros of a polynomial $p(z)$ in an interval $[a, b]$, $-\infty \leq a < b \leq \infty$, can be obtained from a **Sturm sequence**[188] for $p(z)$.

Definition 6.5.1.

A sequence of real polynomials $p_0(x), p_1(x), \ldots, p_m(x)$ 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 some $j < m$, 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., $p_0(x) = q_1(x)p_1(x) - p_2(x)$,

[188] 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. In 1839 he succeeded Poisson in the chair of mechanics at the École Polytechnique, Paris.

where the degree of $p_2(x)$ is strictly less than that of $p_1(x)$. Continuing in this way, we compute $p_2(x), \ldots, p_m(x)$ by

$$p_{k+1}(x) = q_k(x)p_k(x) - p_{k-1}(x), \quad k = 1 : m-1, \tag{6.5.28}$$

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 $p_j(r) = 0$, 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 (6.5.28)*, 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 (6.5.28) 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 6.5.5.

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

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

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 6.5.2. 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 be competitive, when only a relatively small number of real roots in a given interval are of interest. Consider a real symmetric tridiagonal

Table 6.5.2. *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	

matrix,

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

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

$$p_1(\lambda) = \alpha_1 - \lambda, \quad p_k(\lambda) = (\alpha_k - \lambda)p_{k-1}(\lambda) - \beta_k^2 p_{k-2}(\lambda), \quad k = 2 : n. \tag{6.5.29}$$

Then the sequence

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

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. It can also be used for determining the singular values of a bidiagonal matrix in a given interval; see Volume II.

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 [372] and illustrated in Figure 6.5.1, is such a "two-dimensional bisection algorithm."[189] 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 an **exclusion test** applied to squares in the complex plane. Assume that $f(z)$ is analytic in K and that

$$|f'(z)| \leq M \quad \forall\ z \in K.$$

Then if $|f(z_0)| > \eta M$ there can be no zero of $f(z)$ in K. Any square that does not pass this exclusion test may contain a root and 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

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

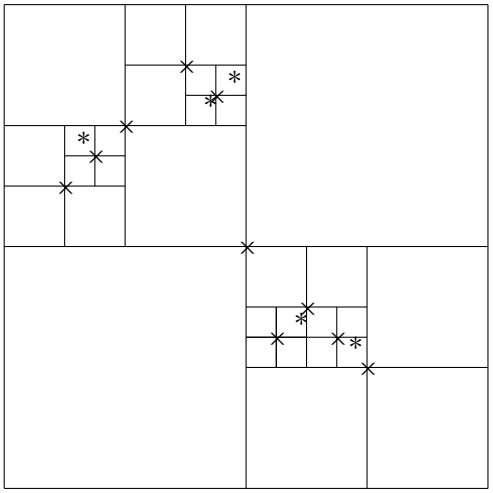

Figure 6.5.1. *Suspect squares computed by Weyl's quadtree method. Their centers (marked by $\times$) approximate the five zeros marked by $*$.*

$p(x)$. 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 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 $1/2$.

6.5.7 Finding Greatest Common Divisors

A problem that arises in many applications, including computer graphics, geometric modelling, and control theory, is the computation of the **greatest common divisor** (GCD) of two polynomials:

$$p(z) = a_0 \prod_{j=1}^{m} (z - \alpha_i) = a_0 z^n + a_1 z^{n-1} + \cdots + a_n, \quad a_0 \neq 0,$$

$$q(z) = b_0 \prod_{j=1}^{m} (z - \beta_j) = b_0 z^m + b_1 z^{n-1} + \cdots + b_m, \quad b_0 \neq 0.$$

The GCD can in principle be determined by the Euclidean algorithm; see Problem 1.2.6. Assume that $n > m$ and perform the divisions

$$p(z) = q(z)s(z) + r_1(z),$$
$$q(z) = r_1(z)s_1(z) + r_2(z),$$

$$\begin{aligned} q(z) &= r_1(z)s_1(z) + r_2(z), \\ &\vdots \\ r_{s-2}(z) &= r_{s-1}(z)s_{s-1}(z) + r_s(z). \end{aligned} \tag{6.5.30}$$

The degrees of the polynomials r_i, $i = 1 : s$, decrease at each step and we stop when r_s is a constant. If $p(z)$ and $q(z)$ have a common factor, then that factor is a divisor of r_i, $i = 1 : s$, as well. This means that if $r_s \neq 0$, then $p(z)$ and $q(z)$ have no common divisor; otherwise r_{s-1} is the greatest common divisor.

From the representation of $r_s(z)$ in (6.5.30) it follows that if $p(z)$ and $q(z)$ are two polynomials, then there always exist polynomials $x(z)$ and $y(z)$ such that $xp - yq$ equals a constant. If $p(z)$ and $q(z)$ have a common divisor, then $xp(z) - yq(z) = 0$.

The Euclidean algorithm is suitable for symbolic computation, but not effective in finite precision arithmetic because of roundoff. We now describe a suitable algorithm for determining the GCD when the coefficients of $p(z)$ and $q(z)$ are given as floating-point numbers. Note that in this case we can only hope to determine "almost common factors."

Consider the product of the mn factors,

$$\bar{R}(p, q) = \prod_{i=1}^{n} \prod_{j=1}^{m} (\beta_j - \alpha_i), \tag{6.5.31}$$

of all possible differences between the two sets of roots. This is a symmetric function of the α's and β's and hence expressible as a polynomial in the quotients a_i/a_0 and b_j/b_0. Clearly $\bar{R}(p, q)$ is zero if and only if $p(z)$ and $q(z)$ have a common zero, and no polynomial of lower degree in the α's and β's can have this property. The polynomial $R(p, q) = a_0^m b_0^n \bar{R}(p, q)$ is called the **resultant** of $p(z)$ and $q(z)$ and can also be expressed as

$$R(p, q) = a_0^m b_0^n \bar{R}(p, q) \equiv a_0^m \prod_{i=1}^{n} q(\alpha_i) \equiv b_0^n \prod_{j=1}^{m} p(\beta_j). \tag{6.5.32}$$

From the coefficients of $p(z)$ and $p(z)$ we form the **Sylvester matrix**[190]

$$S(p, q) = \begin{pmatrix} T_m(p) \\ T_n(q) \end{pmatrix} \in \mathbf{R}^{(m+n)\times(m+n)}, \tag{6.5.33}$$

where

$$T_m(p) = \begin{pmatrix} a_0 & a_1 & \cdots & a_{n-1} & a_n & & \\ & a_0 & a_1 & \cdots & a_{n-1} & a_n & \\ & & \ddots & \ddots & \ddots & \ddots & \\ & & & a_0 & a_1 & \cdots & a_{n-1} & a_n \end{pmatrix} \in \mathbf{R}^{m\times(m+n)},$$

$$T_n(q) = \begin{pmatrix} b_0 & b_1 & \cdots & b_{m-1} & b_m & & \\ & b_0 & b_1 & \cdots & b_{m-1} & b_m & \\ & & \ddots & \ddots & \ddots & \ddots & \\ & & & b_0 & b_1 & \cdots & b_{m-1} & b_m \end{pmatrix} \in \mathbf{R}^{n\times(m+n)}.$$

[190] The Sylvester matrix appeared in [346].

Note the Toeplitz structure of the two blocks of rows. It can be shown that *the resultant* $R(p,q)$ *equals the determinant of* $S(p,q)$ *except for a possible multiplier* -1. Thus $p(z)$ and $q(z)$ have a common zero if and only if the Sylvester matrix is singular.

If we form $m+n$ polynomials of degree less than or equal to $m+n-1$ by multiplying $p(z)$ with z^k, $k = 1:m$, and multiplying $q(z)$ with z^k, $k = 1:n$, this can be written in matrix notation as

$$S(p,q)\,(\,z^{m+n-1} \quad \cdots \quad z \quad 1\,)^T = \begin{pmatrix} f_p \\ f_q \end{pmatrix},$$

where

$$f_p = p(z)\,(\,z^{m-1} \quad \cdots \quad z \quad 1\,)^T, \qquad f_q = q(z)\,(\,z^{n-1} \quad \cdots \quad z \quad 1\,)^T.$$

If γ is a common root to the two polynomials $p(z)$ and $q(z)$, then

$$S(p,q)(\gamma^{m+n-1}, \ldots, \gamma, 1)^T = 0.$$

In other words this vector belongs to the nullspace of $S(p,q)$. The degree of the GCD of $p(z)$ and $q(z)$ equals the dimension of the nullspace of the Sylvester matrix, i.e.,

$$\deg(\gcd(p,q)) = m + n - r, \quad r = \operatorname{rank}(S(p,q)).$$

The best way to estimate the rank is by computing the SVD of the Sylvester matrix; see Theorem 1.4.4. The coefficients of the GCD can also be obtained from the decomposition.

Review Questions

6.5.1 Describe the method of iterated successive synthetic division for computing function values and derivatives of a polynomial.

6.5.2 Consider the polynomial $p(z) = z^4 - 2z^3 - 4z^2 + z + 1$. Using Descartes's rule of sign, what can you deduce about the number of real positive roots?

6.5.3 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.

6.5.4 Discuss the ill-conditioning of roots of polynomial equations. What famous polynomial did Wilkinson use as an example?

6.5.5 (a) What is the companion matrix of a polynomial $p(x) = x^n + a_1 x^{n-1} + \cdots + a_{n-1}x + a_n$?

(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?

6.5.6 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 Euclidean algorithm.

Problems and Computer Exercises

6.5.1 Apply Newton's method to determine one of the complex roots of the equation $z^2+1 = 0$. Start with $z_0 = 1 + i$.

6.5.2 Consider a polynomial with real coefficients

$$p(z) = a_0 z^n + a_1 z^{n-1} + \cdots + a_n, \quad a_i \neq 0, \quad i = 0 : n.$$

(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

$$d(z) = (z - z_0)(z - \bar{z}_0) = z^2 - 2x_0 z + (x_0^2 + y_0^2).$$

Derive a recursion for computing the quotient polynomial $q(z)$ and $p(z_0) = b_{n-1}z_0 + b_n$, where

$$q(z) = b_0 z^{n-2} + b_1 z^{n-3} + \cdots + b_{n-2},$$
$$p(z) = q(z)d(z) + b_{n-1}z + b_n.$$

Count the number of real additions and multiplications needed to compute $p(z_0)$ and also show how to compute $p'(z_0)$.

6.5.3 (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 $x_0 = 2$.

6.5.4 Let C be the companion matrix of $p(x)$. Show that in Bernoulli's method the vector sequence $m_n = (y_{n+k}, \ldots, y_{n+1}, y_n)^T$ can be generated by forming successive matrix-vector products $m_n = Cm_{n-1}$. Conclude that $m_k = C^k m_0$.

6.5.5 (a) Assume that the zeros of a polynomial satisfy the stronger conditions $|u_1| > |u_2| > |u_3| \geq \cdots \geq |u_k|$, where u_1 and u_2 are real and simple. Show that in Bernoulli's method

$$\lim_{n\to\infty} D_n/D_{n-1} = u_1 u_2, \quad D_n = \begin{vmatrix} y_n & y_{n+1} \\ \Delta y_n & \Delta y_{n+1} \end{vmatrix}. \tag{6.5.34}$$

(b) The result in (a) can be combined with (6.5.18) to also compute u_2. What is the rate of convergence in this process?

(c) Apply Bernoulli's method to the polynomial p_4 in Example 6.5.2 to improve the approximations computed by Graeffe's method.

6.5.6 (a) Use the MATLAB command `x = roots(poly(1:20))` to compute the roots of the classical Wilkinson polynomial $\prod_{k=1}^{20}(x-k)$. Use IEEE double precision. What is the maximum error in the roots?

(b) Using the command `poly(x)` to compute the coefficients of the polynomial with the erroneous roots from (a). What is the maximum error in these coefficients?

6.5.7 (a) Generate the coefficients of the Legendre polynomials $P_n(x)$, $n = 2:24$, using the three-term recursion $P_0(x) = 1$, $P_1(x) = x$,

$$P_{n+1}(x) = \frac{2n+1}{n+1} x P_n(x) - \frac{n}{n+1} P_{n-1}(x), \quad n \geq 1,$$

where the recursion coefficients are exact rational numbers. Compute using IEEE double precision the roots of $P_{24}(x)$ as in Problem 6.5.6. Verify that the errors in some computed roots λ_k^* close to 1 exceed $10^6 u$, where $u = 1.11\,10^{-16}$ is the unit roundoff.

(b) Compute the coefficients of the polynomial $\prod_{k=1}^{24}(x-\lambda^*)$. Show that the relative errors in the computed coefficients are all less than $16u$.

6.5.8 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, for example, 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** known 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 $-2.25 \leq \Re c \leq 0.75$, $-1.5 \leq \Im c \leq 1.5$.

6.5.9 The following MATLAB function, `sylvester`, generates the Sylvester matrix of two polynomials whose coefficients are given in the row vectors a and b:

```
function S = sylvest(a,b);
% SYLVEST computes
n = length(a) - 1;
m = length(b) - 1;
r = [a,zeros(1,m-1)];
c = [a(1),zeros(1,m-1)];
T1 = toeplitz(c,r);
r = [b,zeros(1,n-1)];
c = [b(1),zeros(1,n-1)];
T2 = toeplitz(c,r);
S = [T1; T2];
```

Generate $S(p, q)$ for the two polynomials

$$p(x) = x^5 + x^4 + x^3 + x^2 + x + 1,$$
$$q(x) = x^4 - 2x^3 + 3x^2 - x - 71.$$

The GCD of these polynomials is $x + 1$. How is this revealed by the SVD of $S(p, q)$?
(b) Try some more difficult examples with several common zeros.

Notes and References

The general idea of solving an equation by repeatedly improving an estimate of the solution has been used in many cultures for thousands of years. Examples are ancient Greek and Babylonian methods as well as Arabic algebraists from the eleventh century.

An interesting historical account of Newton's method is given in Ypma [388]. Newton's method is contained in his book *Method of Fluxions*, written in 1671 but not published until 1736. Joseph Raphson was allowed to see Newton's work and Newton's method 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. Edmond Halley was a contemporary of Isaac Newton and his third-order method was published more than 300 years ago [179].

Several comprehensive monographs dealing with methods for solving scalar nonlinear equations are available. Traub [354] 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 [279]. The recently reprinted book by Brent [44] deals exclusively with methods which only use function values for finding zeros and minima of functions of a single variable. It is unique in its careful treatment of algorithmic details which are crucial when developing reliable computer codes.

Fortran and C versions of some of the zero-finding and minimization routines given in [44] are available from Netlib. The MATLAB function `fminbnd` is based on the Fortran implementation FMIN of Brent's algorithm given in Forsythe, Malcolm, and Moler [123, pp.184–187].

Halley's method has been rediscovered by J. H. Lambert [234] and numerous other people; see Traub [354, Sec. 5.22]. A nice exposition of this and other third-order methods is given by Gander [129]. Families of iteration methods of arbitrary order are studied in a remarkable paper by Schröder [316], which has been translated into English by G. W. Stewart [317]. 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 [356]; see also [204, Sec. 4.4]. This family was derived independently by Kalantari, Kalantari, and Zaare-Nahandi [221].

There is a vast literature on methods for the classical problem of solving algebraic equations. Indeed, the nonexistence of an algebraic formula for equations of fifth and higher degree gave rise to modern algebra. Householder [204] gives an elegant treatment and is an excellent source for classical results. Detailed surveys are found also in Durand [102] (in French) and Sendov, Andreev, and Kjurkchiev [320]. Numerical polynomial algebra is an

emerging area that falls between numerical analysis and computer algebra. A comprehensive survey of this area is given by Stetter [334]. Further studies of the speed and accuracy of computing polynomial zeros by the MATLAB method of solving the eigenvalue problem for the companion matrix are given in [157, 104].

The theory of Sturm sequences is treated in [204, Sec. 2.5]. The quadtree method was used by Weyl [372] 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 [282], who also gives a brief account of the history of algorithms for solving polynomial equations. Some background on algorithms for computing the GCD is found in [2].