

# Chapter 1

## Numerical algorithms

The word ‘algorithm’ derives from the name of the Persian mathematician (Abu Ja’far Muhammad ibn Musa) Al-Khwarizmi who lived from about 790 CE to about 840 CE. He wrote a book *Hisab al-jabr w’al-muqabala* that also named the subject ‘algebra.’

Numerical analysis is the subject which studies algorithms for computing expressions defined with real numbers. There are two different phases to address:

- the development of algorithms, and
- the analysis of algorithms.

These are in principle independent activities, but in reality the development of an algorithm is often guided by the analysis of the algorithm, or of a simpler algorithm that computes the same thing.

There are three characteristics of algorithms using real numbers that are in conflict to some extent:

- the accuracy (or consistency) of the algorithm, and
- the stability of the algorithm.
- the effects of finite precision arithmetic.

The first of these just means that the algorithm approximates the desired quantity to any required accuracy under suitable restrictions. The second means that the behavior of the algorithm is continuous with respect to the parameters of the algorithm. The latter topic is still not well understood at the most basic level. We will see that in trying to improve the accuracy or efficiency of a stable algorithm one is often led into considering algorithms that turn out to be unstable, and therefore of minimal (if any) value. These three aspects of numerical analysis are often intertwined, as

ultimately we want an algorithm that we can analyze to prove it is effective when using computer arithmetic.

The **efficiency** of an algorithm is a more complex concept, but is often the bottom line in choosing one algorithm over another. It can be related to all of the above characteristics, as well as the **complexity** of the algorithm in terms of computational work or memory references required in its implementation.

We begin with a problem from antiquity to illustrate each of these components of numerical analysis in an elementary context. We will not always disentangle the different issues, but we hope that the differing components will be evident.

## 1.1 Finding roots

People have been computing roots for millennia. Evidence exists [21] that the Babylonians, who used base-60 arithmetic, were able to approximate

$$\sqrt{2} \approx 1 + \frac{24}{60} + \frac{51}{60^2} + \frac{10}{60^3} \quad (1.1)$$

nearly four thousand years ago. By the time of Heron<sup>1</sup> a method to compute square roots was established [10] that we recognize now as the Newton-Raphson method (see Section 2.2.1) and takes the form of a repeated iteration

$$x \leftarrow \frac{1}{2}(x + y/x) \quad (1.2)$$

where the backwards arrow  $\leftarrow$  means **assignment** in algorithms. That is, once the computation of the expression on the right-hand side of the arrow has been completed, a new value is assigned to the variable  $x$ .

The algorithm (1.2) is an example of what is known as **fixed-point iteration**, in which one hopes to find a fixed point, that is, an  $x$  where the iteration quits changing. A **fixed point** is thus a point  $x$  where

$$x = \frac{1}{2}(x + y/x). \quad (1.3)$$

More precisely,  $x$  is a fixed point of the function

$$f(x) = \frac{1}{2}(x + y/x), \quad (1.4)$$

defined, say, for  $x \neq 0$ . If we re-arrange terms in (1.3), we find  $x = y/x$ , or  $x^2 = y$ . Thus a fixed point as defined in (1.3) is a solution to  $x^2 = y$ , so that  $x = \pm\sqrt{y}$ .

To describe actual implementations of these algorithms, we choose Matlab's scripting syntax. As a programming language, this has some limitations, but its use is extremely widespread. In addition to the commercial interpreter provided by the Mathworks company, a public domain implementation called **octave** is available.

We can implement (1.2) in Matlab/octave in two steps as follows. First, we define the function (1.4) via the code

---

<sup>1</sup>A.k.a. Hero, of Alexandria, who lived in the first century CE.

$\sqrt{2}$ approximation	absolute error
1.500000000000000	8.5786e-02
1.41 <b>6</b> 666666666667	2.4531e-03
1.41421 <b>5</b> 68627451	2.1239e-06
1.41421356237 <b>4</b> 69	1.5947e-12
1.41421356237309	-2.2204e-16

Table 1.1: Results of experiments with the Heron algorithm applied to approximate  $\sqrt{2}$  using the algorithm (1.2) starting with  $x = 1$ . The boldface indicates the leading incorrect digit. Note that the number of correct digits essentially doubles at each step.

```
function x=heron(x,y)
x=.5*(x+y/x);
```

To use this function, you need to start with some initial guess, say  $x = 1$ , which is written simply as

```
x=1
```

(Writing an expression with and without a semi-colon at the end controls whether the interpreter prints the result or not.) But then you simply iterate:

```
x=heron(x,y)
```

until  $x$  (or the part you care about) quits changing.

We can examine the accuracy by a simple code

```
function x=errheron(x,y)
for i=1:5
    x=heron(x,y);
    errheron=x-sqrt(y)
end
```

We show in Table 1.1 the results of these computations in the case  $y = 2$ . This algorithm seems to be quite magic because it ‘homes in’ on the solution. We will see that the accuracy is doubling at each step.

### 1.1.1 Relative versus absolute error

We can require the accuracy of an algorithm to be based on the size of the answer. For example, we might want the approximation  $\hat{x}$  of a root  $x$  to be small relative to the size of  $x$ :

$$\frac{\hat{x}}{x} = 1 + \delta, \quad (1.5)$$

where  $\delta$  satisfies some fixed tolerance, e.g.,  $|\delta| \leq \epsilon$ . Such a requirement is in keeping with the model we will adopt for floating point operations (see (1.29) and Section 14.1).

We can examine the relative accuracy by the simple code

```
function x=relerrher(x,y)
for i=1:6
    x=heron(x,y);
    errheron=(x/sqrt(y))-1
end
```

We leave as Exercise 1.2 to compare the results produced by the code `relerrher` with the absolute errors presented in Table 1.1.

### 1.1.2 Scaling Heron's algorithm

Before we analyze how Heron's algorithm (1.2) works, let us enhance it by a pre-scaling. To begin with, we can suppose that the number  $y$  whose square root we seek lies in the interval  $[\frac{1}{2}, 2]$ . If  $y < \frac{1}{2}$  or  $y > 2$ , then we make the transformation

$$\tilde{y} = 4^k y \quad (1.6)$$

to get  $\tilde{y} \in [\frac{1}{2}, 2]$ , for some integer  $k$ . And of course  $\sqrt{\tilde{y}} = 2^k \sqrt{y}$ . By scaling  $y$  in this way, we limit the range of inputs that the algorithm must deal with.

In Table 1.1, we showed the absolute error for approximating  $\sqrt{2}$ , and in Exercise 1.2 the relative errors for approximating  $\sqrt{2}$  and  $\sqrt{\frac{1}{2}}$  are explored. It turns out that the maximum errors for the interval  $[\frac{1}{2}, 2]$  occur at the ends of the interval (Exercise 1.3). Thus five iterations of Heron are sufficient to compute  $\sqrt{y}$  to sixteen decimal places.

### 1.1.3 Analyzing Heron's algorithm

We can write an algebraic expression linking the error at one iteration to the error at the next. Thus define

$$x_{n+1} = \frac{1}{2}(x_n + y/x_n) \quad (1.7)$$

and let  $e_n = x_n - x = x_n - \sqrt{y}$ . Then by (1.7) and (1.3),

$$\begin{aligned} e_{n+1} &= x_{n+1} - x = \frac{1}{2}(x_n + y/x_n) - \frac{1}{2}(x + y/x) = \frac{1}{2}(e_n + y/x_n - y/x) \\ &= \frac{1}{2} \left( e_n + \frac{y(x - x_n)}{xx_n} \right) = \frac{1}{2} \left( e_n - \frac{xe_n}{x_n} \right) = \frac{1}{2} e_n \left( 1 - \frac{x}{x_n} \right) = \frac{1}{2} \frac{e_n^2}{x_n}. \end{aligned} \quad (1.8)$$

If we are interested in the relative error,

$$\hat{e}_n = \frac{e_n}{x} = \frac{x_n - x}{x} = \frac{x_n}{x} - 1, \quad (1.9)$$

then (1.8) becomes

$$\hat{e}_{n+1} = \frac{1}{2} \frac{x \hat{e}_n^2}{x_n} = \frac{1}{2} (1 + \hat{e}_n)^{-1} \hat{e}_n^2. \quad (1.10)$$

Thus we see that the error at each step is proportional to the square of the error at the previous step; for the relative error, the constant of proportionality tends rapidly to  $\frac{1}{2}$ . In addition, (1.10) implies a limited type of **global convergence** property, at least for  $x_n > \alpha$ . In that case, (1.10) gives

$$|\hat{e}_{n+1}| = \frac{1}{2} \frac{\hat{e}_n^2}{|1 + \hat{e}_n|} = \frac{1}{2} \frac{\hat{e}_n^2}{1 + \hat{e}_n} \leq \frac{1}{2} \hat{e}_n. \quad (1.11)$$

Thus the relative error gets reduced by a factor smaller than one half at each iteration, no matter how large the initial error may be. Unfortunately, this type of global convergence property does not hold for many algorithms. We can illustrate what can go wrong in the case of the Heron algorithm when  $x_n < \alpha$ .

Suppose for simplicity that  $y = 1$ , so that also  $x = 1$ , so that the relative error is  $\hat{e}_n = x_n - 1$ , and therefore (1.10) implies that

$$\hat{e}_{n+1} = \frac{1}{2} \frac{(1 - x_n)^2}{x_n} \quad (1.12)$$

As  $x_n \rightarrow 0$ ,  $\hat{e}_{n+1} \rightarrow \infty$ , even though  $|\hat{e}_n| < 1$ . Therefore, convergence is not truly global for the Heron algorithm.

## 1.2 Where to start

With any iterative algorithm, we have to start the iteration somewhere, and this choice can be an interesting problem in its own right. Just like the initial scaling described in Section 1.1.2, this can affect the performance of the overall algorithm substantially.

For the Heron algorithm, there are various possibilities. The simplest is just to take  $x_0 = 1$ , in which case

$$\hat{e}_0 = \frac{1}{x} - 1. \quad (1.13)$$

This gives

$$\hat{e}_1 = \frac{1}{2} x \hat{e}_0^2 = \frac{1}{2} x \left( \frac{1}{x} - 1 \right)^2 = \frac{1}{2} \frac{(x - 1)^2}{x}. \quad (1.14)$$

If we think of  $\hat{e}_1$  as a function of  $x$  (it is by definition a function of  $y = x^2$ ), then we see that

$$\hat{e}_1(x) = \hat{e}_1(1/x). \quad (1.15)$$

Note that the maximum of  $\hat{e}_1(x)$  on  $[2^{-1/2}, 2^{1/2}]$  occurs at the ends of the interval, and

$$\hat{e}_1(\sqrt{2}) = \frac{1}{2} \frac{(\sqrt{2} - 1)^2}{\sqrt{2}} = \frac{3}{4} \sqrt{2} - 1 \approx 0.060660. \quad (1.16)$$

Thus the simple starting value  $x_0 = 1$  is remarkably effective. Nevertheless, let us see if we can do better.

### 1.2.1 Another start

Another idea to start the iteration is to make an approximation to the square root function given the fact that we always have  $y \in [\frac{1}{2}, 2]$  (Section 1.1.2). Since this means that  $y$  is near one, we can write  $y = 1 + t$  (i.e.,  $t = y - 1$ ), and we get that

$$\begin{aligned} x = \sqrt{y} &= \sqrt{1+t} = 1 + \frac{1}{2}t + \mathcal{O}(t^2) \\ &= 1 + \frac{1}{2}(y-1) + \mathcal{O}(t^2) = \frac{1}{2}(y+1) + \mathcal{O}(t^2). \end{aligned} \quad (1.17)$$

Thus we get the approximation  $x \approx \frac{1}{2}(y+1)$  as a possible starting guess:

$$x_0 = \frac{1}{2}(y+1). \quad (1.18)$$

But this is the same as  $x_1$  if we had started with  $x_0 = 1$ . Thus we have not really gotten anything new.

### 1.2.2 The best start

Our first attempt (1.18) based on a linear approximation to the square root did not produce a new concept, since it gives the same result as starting with a constant guess after one iteration. The approximation (1.18) corresponds to the tangent line of the graph of  $\sqrt{y}$  at  $y = 1$ , but this may not be the best linear approximation to a function on an interval. So let us ask the question: what is the best approximation to  $\sqrt{y}$  on the interval  $[\frac{1}{2}, 2]$ ? This problem is a miniature of the questions we will address in Chapter 11.

The general linear function is of the form

$$f(y) = a + by. \quad (1.19)$$

If we take  $x_0 = f(y)$ , then the relative error  $\hat{e}_0$  is

$$\hat{e}_0 = \frac{x_0 - \sqrt{y}}{\sqrt{y}} = \frac{a + by - \sqrt{y}}{\sqrt{y}} = \frac{a}{\sqrt{y}} + b\sqrt{y} - 1. \quad (1.20)$$

Let us write  $e_{ab}(y) = \hat{e}_0$  to be precise. We seek  $a$  and  $b$  such that the maximum of  $|e_{ab}|$  over  $[\frac{1}{2}, 2]$  is minimized.

Fortunately, the functions

$$e_{ab}(y) = \frac{a}{\sqrt{y}} + b\sqrt{y} - 1 \quad (1.21)$$

have a simple structure. As always, it is helpful to compute the derivative:

$$e'_{ab}(y) = -\frac{1}{2}ay^{-3/2} + \frac{1}{2}by^{-1/2} = \frac{1}{2}(-a + by)y^{-3/2}. \quad (1.22)$$

Thus  $e'_{ab}(y) = 0$  for  $y = a/b$ ; further,  $e'_{ab}(y) > 0$  for  $y > a/b$ , and  $e'_{ab}(y) < 0$  for  $y < a/b$ . Therefore,  $e_{ab}$  has a minimum at  $y = a/b$  and is strictly increasing as we move away from that point in either direction. Thus the maximum values of  $|e_{ab}|$  on  $[\frac{1}{2}, 2]$  will be at the ends of the interval or at  $y = a/b$  if  $a/b \in [\frac{1}{2}, 2]$ . Moreover, the best value of  $e_{ab}(a/b)$  will be negative (Exercise 1.9). Thus we consider the three values

$$\begin{aligned} e_{ab}(2) &= \frac{a}{\sqrt{2}} + b\sqrt{2} - 1 \\ e_{ab}(\frac{1}{2}) &= a\sqrt{2} + \frac{b}{\sqrt{2}} - 1 \\ -e_{ab}(a/b) &= 1 - 2\sqrt{ab} \end{aligned} \quad (1.23)$$

Note that  $e_{ab}(2) = e_{ba}(1/2)$  and  $\min e_{ab} = \min e_{ba} = 1 - 2\sqrt{ab}$ . Therefore, the optimal values of  $a$  and  $b$  must be the same:  $a = b$  (Exercise 1.10). Moreover, the minimum value of  $e_{ab}$  must be minus the maximum value on the interval (Exercise 1.11). Thus the optimal value of  $a = b$  is characterized by

$$a\frac{3}{2}\sqrt{2} - 1 = 1 - 2a \implies a = \left(\frac{3}{4}\sqrt{2} + 1\right)^{-1}. \quad (1.24)$$

Recall that the simple idea of starting the Heron algorithm with  $x_0 = 1$  yielded an error  $|\hat{e}_1| \leq \gamma = \frac{3}{4}\sqrt{2} - 1$ , and that this was equivalent to choosing  $a = \frac{1}{2}$  in the current scheme. Note that the optimal  $a = 1/(\gamma + 2)$ , only slightly less than a half, and the resulting minimum value of the maximum of  $e_{aa}$  is

$$1 - 2a = 1 - \frac{2}{\gamma + 2} = \frac{\gamma}{\gamma + 2}. \quad (1.25)$$

Thus the optimal value of  $a$  reduces the previous error of  $\gamma$  (for  $a = \frac{1}{2}$ ) by nearly a factor of a half, despite the fact that the change in  $a$  is quite small. The benefit of using the better initial guess is of course squared at each iteration, so the reduced error is nearly smaller by a factor of  $2^{-2^k}$  after  $k$  iterations of Heron.

### 1.3 An unstable algorithm

Heron's algorithm has one drawback in that it requires division. One can imagine that a simpler algorithm might be possible such as

$$x \leftarrow x + x^2 - y \quad (1.26)$$

$n$	0	1	2	3	4	5
$x_n$	1.5	1.75	2.81	8.72	82.8	6937.9
$n$	6	7	8	9	10	11
$x_n$	$4.8 \times 10^7$	$2.3 \times 10^{15}$	$5.4 \times 10^{30}$	$2.9 \times 10^{61}$	$8.3 \times 10^{122}$	$6.9 \times 10^{245}$

Table 1.2: Unstable behavior of the iteration (1.26) for computing  $\sqrt{2}$ .

Before experimenting with this algorithm, we note that a fixed point

$$x = x + x^2 - y \tag{1.27}$$

does have the property that  $x^2 = y$ , as desired. Thus we can assert the *accuracy* of the algorithm (1.26), in the sense that any fixed point will solve the desired problem. However, it is easy to see that the algorithm is not **stable** in the sense that if we start with an initial guess with any sort of error, the algorithm fails. Table 1.2 shows the results of applying (1.26) starting with  $x_0 = 1.5$ . What we see is a rapid movement *away* from the solution, followed by a catastrophic blow-up (which eventually causes failure in a fixed precision arithmetic system, or causes the computer to run out of memory in a variable precision system). The error is again being squared, as with the Heron algorithm, but since the error is getting bigger rather than smaller, the algorithm is useless. We will see how to diagnose instability (or rather how to guarantee stability) for iterations like (1.26) in Section 2.1.

## 1.4 General roots: effects of floating point

So far, we have seen no adverse effects related to finite precision arithmetic. This is common for (stable) iterative methods like the Heron algorithm. But now we consider a more complex problem in which rounding plays a dominant role.

Suppose that we want to compute the roots of a general quadratic equation  $x^2 + 2bx + c = 0$  where  $b < 0$ , and we chose the algorithm

$$x \leftarrow -b + \sqrt{b^2 - c} \tag{1.28}$$

Note that we have assumed that we can compute the square root function as part of this algorithm, say by Heron’s method.

Unfortunately, the simple algorithm in (1.28) fails if we have  $c = \epsilon^2 b^2$  (it returns  $x = 0$ ) as soon as  $\epsilon^2 = c/b^2$  is small enough that the floating point representation of  $1 - \epsilon^2$  is 1. For any (fixed) finite representation of real numbers, this will occur for some  $\epsilon > 0$ .

We will consider floating point arithmetic in more detail in Section 14.1, but the simple model we adopt says that the result of computing a binary operator  $\oplus$  such

as  $+$ ,  $-$ ,  $/$ , or  $*$  has the property that

$$fl(a \oplus b) = (a \oplus b)(1 + \delta). \quad (1.29)$$

where  $|\delta| \leq \epsilon$  where  $\epsilon > 0$  is a parameter of the model.<sup>2</sup> However, this means that a collection of operations could lead to catastrophic cancellation, e.g.,  $fl(fl(1 + \frac{1}{2}\epsilon) - 1) = 0$  and not  $\frac{1}{2}\epsilon$ .

We can see the behavior in some simple codes. But first, let us simplify the problem further so that we just have one parameter to deal with. Suppose that the equation to be solved is of the form

$$x^2 - 2bx + 1 = 0. \quad (1.30)$$

That is, we switch  $b$  to  $-b$  and set  $c = 1$ . In this case, the two roots are multiplicative inverses of each other. Define

$$x_{\pm} = b \pm \sqrt{b^2 - 1}. \quad (1.31)$$

Then  $x_- = 1/x_+$ .

There are various possible algorithms. We could use one of the two formulae  $x_{\pm} = b \pm \sqrt{b^2 - 1}$  directly. More precisely, let us write  $\tilde{x}_{\pm} \approx b \pm \sqrt{b^2 - 1}$  to indicate that we implement this in floating point. Correspondingly, there is another pair of algorithms that start by computing  $\tilde{x}_{\mp}$  and then define, say,  $\hat{x}_+ \approx 1/\tilde{x}_-$ . A similar algorithm could determine  $\hat{x}_- \approx 1/\tilde{x}_+$ .

All four of these algorithms will have different behaviors. We expect that the behaviors of the algorithms for computing  $\tilde{x}_-$  and  $\hat{x}_-$  will be dual in some way to those for computing  $\tilde{x}_+$  and  $\hat{x}_+$ , so we consider only the first pair.

First of all, the function `minus` implements the  $\tilde{x}_-$  square root algorithm:

```
function x=minus(b)
% solving = 1-2bx +x^2
x=b-sqrt(b^2-1);
```

To know if it is getting the right answer, we need another one to check the answer:

```
function error=check(b,x)
error = 1-2*b*x +x^2;
```

To automate the process, we put the two together:

```
function error=chekminus(b)
x=minus(b);
error=check(b,x)
```

---

<sup>2</sup>The notation  $fl$  is somewhat informal. It might be more precise to write  $af\ell(\oplus)b$  instead of  $fl(a \oplus b)$ , since the operator is modified by the effect of rounding.

For example, when  $b = 10^6$ , we find the error is  $-7.6 \times 10^{-6}$ . As  $b$  increases further, the error increases, ultimately leading to complete nonsense. For this reason, we consider an alternative algorithm suitable for large  $b$ .

The algorithm for  $\hat{x}_-$  is given by

```
function x=plusinv(b)
% solving = 1-2bx +x^2
y=b+sqrt(b^2-1);
x=1/y;
```

Similarly, we can check the accuracy of this computation by the code

```
function error=chekplusinv(b)
x=plusinv(b);
error=check(b,x)
```

Now when  $b = 10^6$ , we find the error is  $-2.2 \times 10^{-17}$ . And the bigger  $b$  becomes, the more accurate it becomes.

Here we have seen that algorithms can have data-dependent behavior with regard to the effects of finite-precision arithmetic.

## 1.5 Exercises

**Exercise 1.1** *How accurate is the approximation (1.1) if it is expressed as a decimal approximation (how many digits are correct)?*

**Exercise 1.2** *Run the code `relerrher` starting with  $x = 1$  and  $y = 2$  to approximate  $\sqrt{2}$ . Compare the results with Table 1.1. Also run the code with  $x = 1$  and  $y = \frac{1}{2}$  and compare the results with the previous case. Explain what you find.*

**Exercise 1.3** *Show that the maximum relative error in Heron's algorithm for approximating  $\sqrt{y}$  for  $y \in [1/M, M]$ , for a fixed number of iterations and starting with  $x_0 = 1$ , occurs at the ends of the interval:  $y = 1/M$  and  $y = M$ . (Hint: consider (1.10) and (1.14) and show that the function  $\phi(x) = \frac{1}{2}(1+x)^{-1}x^2$  plays a role in each. Show that  $\phi$  is increasing on the interval  $[0, \infty[$ .)*

**Exercise 1.4** *It is sometimes easier to demonstrate the relative accuracy of an approximation  $\hat{x}$  to  $x$  by showing that*

$$|x - \hat{x}| \leq \epsilon' |\hat{x}| \tag{1.32}$$

*instead of verifying (1.5) directly. Show that if (1.32) holds, then (1.5) holds with  $\epsilon = \epsilon'/(1 - \epsilon')$ .*

**Exercise 1.5** *There is a simple generalization to Heron's algorithm for finding  $k$ -roots as follows:*

$$x \leftarrow \frac{1}{k}((k-1)x + y/x^{k-1}) \quad (1.33)$$

*Show that, if this converges, it converges to a solution of  $x^k = y$ . Examine the speed of convergence both computationally and by estimating the error algebraically.*

**Exercise 1.6** *Show that the error in Heron's algorithm for approximating  $\sqrt{y}$  satisfies*

$$\frac{x_n - \sqrt{y}}{x_n + \sqrt{y}} = \left( \frac{x_0 - \sqrt{y}}{x_0 + \sqrt{y}} \right)^{2^n} \quad (1.34)$$

*for  $n \geq 1$ . Note that the denominator in the left-hand side of (1.34) converges rapidly to  $2\sqrt{y}$ .*

**Exercise 1.7** *We have implicitly been assuming that we were attempting to compute a positive square root with Heron's algorithm, and thus we always started with a positive initial guess. If we give it zero as initial guess, there is immediate failure due to division by zero. But what happens if we start with a negative initial guess? (Hint: there are usually two roots to  $x^2 = y$ , one of which is negative.)*

**Exercise 1.8** *Consider the iteration*

$$x \leftarrow \frac{3}{2}x - \frac{1}{2}yx^3 \quad (1.35)$$

*and show that, if this converges, it converges to a solution of  $x = y^{-\frac{1}{2}}$ . Note that this does not require a division. Show how this could be used to approximate  $1/y$  without using any division.*

**Exercise 1.9** *Suppose that  $a + by$  is the best linear approximation to  $\sqrt{y}$  in terms of relative error on  $[\frac{1}{2}, 2]$ . Prove that the error expression  $e_{ab}$  has to be negative at its minimum. (Hint: if not, you can always decrease  $a$  to make  $e_{ab}(2)$  and  $e_{ab}(\frac{1}{2})$  smaller without increasing the maximum value of  $|e_{ab}|$ .)*

**Exercise 1.10** *Suppose that  $a + by$  is the best linear approximation to  $\sqrt{y}$  in terms of relative error on  $[\frac{1}{2}, 2]$ . Prove that  $a = b$ .*

**Exercise 1.11** *Suppose that  $a + ay$  is the best linear approximation to  $\sqrt{y}$  in terms of relative error on  $[\frac{1}{2}, 2]$ . Prove that the error expression  $e_{aa}(1) = -e_{aa}(2)$ . (Hint: if not, you can always decrease  $a$  to make  $e_{aa}(2)$  and  $e_{aa}(\frac{1}{2})$  smaller without increasing the maximum value of  $|e_{ab}|$ .)*

**Exercise 1.12** *Change the function `minus` for computing  $\tilde{x}_-$  and the function `plusinv` for computing  $\hat{x}_-$  to functions for computing  $\tilde{x}_+$  (call that function `plus`) and  $\hat{x}_+$  (call that function `minusinv`). Use the `check` function to see where they work well and where they fail. Compare that with the corresponding behavior for `minus` and `plusinv`.*

## 1.6 Solutions

**Solution of Exercise 1.3.** The function  $\phi(x) = \frac{1}{2}(1+x)^{-1}x^2$  is increasing on the interval  $[0, \infty[$  since

$$\phi'(x) = \frac{1}{2} \frac{2x(1+x) - x^2}{(1+x)^2} = \frac{1}{2} \frac{2x + x^2}{(1+x)^2} > 0, \quad (1.36)$$

for  $x > 0$ . The expression (1.10) says that  $\hat{e}_{n+1} = \phi(\hat{e}_n)$ , and (1.14) says that  $\hat{e}_1 = \phi(x-1)$ . Thus  $\hat{e}_2 = \phi(\phi(x-1))$ . By induction, define  $\phi^{[n+1]}(t) = \phi(\phi^{[n]}(t))$ , where  $\phi^{[1]}(t) = \phi(t)$  for all  $t$ . Then, by induction,  $\hat{e}_n = \phi^{[n]}(x-1)$  for all  $n \geq 1$ . Since the composition of increasing functions is increasing, each  $\phi^{[n]}$  is increasing, by induction. Thus  $\hat{e}_n$  is maximized when  $x$  is maximized, at least for  $x > 1$ . Note that  $\phi(x-1) = \phi((1/x)-1)$ , so we may also write  $\hat{e}_n = \phi^{[n]}((1/x)-1)$ . Thus the error is symmetric via the relation  $\hat{e}_n(x) = \hat{e}_n(1/x)$ . Thus the maximal error on an interval  $[1/M, M]$  occurs simultaneously at  $1/M$  and  $M$ .

**Solution of Exercise 1.6.** Define  $d_n = x_n + x$ . Then (1.34) in Exercise 1.6 is equivalent to the statement that

$$\frac{e_n}{d_n} = \left( \frac{e_0}{d_0} \right)^{2^n}. \quad (1.37)$$

Thus we compute

$$\begin{aligned} d_{n+1} &= x_{n+1} + x = \frac{1}{2}(x_n + y/x_n) + \frac{1}{2}(x + y/x) = \frac{1}{2}(d_n + y/x_n + y/x) \\ &= \frac{1}{2} \left( d_n + \frac{y(x+x_n)}{xx_n} \right) = \frac{1}{2} \left( d_n + \frac{yd_n}{xx_n} \right) = \frac{1}{2} \left( d_n + \frac{xd_n}{x_n} \right) \\ &= \frac{1}{2} d_n \left( 1 + \frac{x}{x_n} \right) = \frac{1}{2} d_n \left( \frac{x_n + x}{x_n} \right) = \frac{1}{2} \frac{d_n^2}{x_n}. \end{aligned} \quad (1.38)$$

Recall that (1.8) says that  $e_{n+1} = \frac{1}{2}e_n^2/x_n$ , so dividing by (1.38) yields

$$\frac{e_{n+1}}{d_{n+1}} = \left( \frac{e_n}{d_n} \right)^2 \quad (1.39)$$

for any  $n \geq 0$ . A simple induction on  $n$  yields (1.37) as required.

# Chapter 2

## Nonlinear equations

The word for mathematics in Sanskrit is ... and derives from the words...

We begin with one equation in one variable and later extend to systems in Chapter 8. We will see that the method introduced in (1.2) as Heron's method, namely

$$x_{n+1} = \frac{1}{2} \left( x_n + \frac{y}{x_n} \right), \quad (2.1)$$

can be viewed as Newton's method for computing  $\sqrt{y}$ . This provides a general paradigm for solving nonlinear equations iteratively. But first we develop a general framework for iterative solution methods.

### 2.1 Fixed point iteration

This goes by many names, including functional iteration, but we prefer the term *fixed point iteration*, because it seeks to find a fixed point

$$\alpha = g(\alpha) \quad (2.2)$$

for a continuous function  $g$ . Fixed point iteration

$$x_{n+1} = g(x_n) \quad (2.3)$$

has the important property that, *if it converges, it converges to a fixed point* (2.2) (assuming only that  $g$  is continuous). This result is so simple (see Exercise 2.1) that we hesitate to call it a theorem. But it is really the key fact about fixed point iteration.

The rest of the story about fixed point iteration is then to figure out when and how fast it will converge. For example, if  $g$  is Lipschitz continuous with constant  $\lambda < 1$ , that is,

$$|g(x) - g(y)| \leq \lambda|x - y|, \quad (2.4)$$

then convergence will happen if we start close enough to  $\alpha$ . This is easily proved by defining, as we did for Heron's method,  $e_n = x_n - \alpha$  and estimating

$$|e_{n+1}| = |g(x_n) - g(\alpha)| \leq \lambda|e_n|, \quad (2.5)$$

where the equality results from subtracting (2.2) from (2.3). Thus, by induction

$$|e_n| \leq \lambda^n |e_0| \quad (2.6)$$

for all  $n \geq 1$ . Thus we have proved the following.

**Theorem 2.1** *Suppose that  $\alpha = g(\alpha)$  and that the Lipschitz estimate (2.4) holds with  $\lambda < 1$  for all  $x, y \in [\alpha - A, \alpha + A]$  for some  $A > 0$ . Suppose that  $|x_0 - \alpha| \leq A$ . Then the fixed-point iteration defined in (2.3) converges according to (2.6).*

**Proof.** The only small point to be sure about is that all of the iterates stay in the interval  $[\alpha - A, \alpha + A]$ , but this follows from the estimate (2.6) once we know that  $|e_0| \leq A$ , as we have assumed. **QED**

### 2.1.1 Verifying the Lipschitz condition

A Lipschitz continuous function need not be  $C^1$ , but when a function is  $C^1$ , its derivative gives a good estimate of the Lipschitz constant. We formalize this simple result to highlight the idea.

**Lemma 2.1** *Suppose  $g \in C^1$  in an interval around an arbitrary point  $\alpha$ . Then for any  $\epsilon > 0$  there is an  $A > 0$  such that  $g$  satisfies (2.4) in the interval  $[\alpha - A, \alpha + A]$  with  $\lambda \leq |g'(\alpha)| + \epsilon$ .*

**Proof.** By the continuity of  $g'$ , we can pick  $A > 0$  such that  $|g'(t) - g'(\alpha)| < \epsilon$  for all  $t \in [\alpha - A, \alpha + A]$ . Therefore

$$|g'(t)| \leq |g'(\alpha)| + |g'(t) - g'(\alpha)| < |g'(\alpha)| + \epsilon \quad (2.7)$$

for all  $t \in [\alpha - A, \alpha + A]$ . Let  $x, y \in [\alpha - A, \alpha + A]$ , with  $x \neq y$ . Then

$$\begin{aligned} \left| \frac{g(x) - g(y)}{x - y} \right| &= \left| \frac{1}{x - y} \int_y^x g'(t) dt \right| \\ &\leq \max \{ |g'(t)| \mid t \in [\alpha - A, \alpha + A] \} \\ &\leq |g'(\alpha)| + \epsilon, \end{aligned} \quad (2.8)$$

by using (2.7). **QED**

As a result, we conclude that the condition  $|g'(\alpha)| < 1$  is sufficient to guarantee convergence of fixed-point iteration, as long as we start close enough to the root  $\alpha = g(\alpha)$  (cf. Exercise 2.3).

On the other hand, the Lipschitz constant  $\lambda$  in (2.4) also gives an upper bound for the derivative:

$$|g'(\alpha)| \leq \lambda \quad (2.9)$$

(cf. Exercise 2.4). Thus if  $|g'(\alpha)| > 1$ , fixed point iteration will likely not converge, since the Lipschitz constant for  $g$  will be greater than one in any such interval. If we recall the iteration function  $g(x) = x + x^2 - y$  in (1.26), we see that  $g'(\sqrt{y}) = 1 + 2\sqrt{y} > 1$ . Thus the divergence of that algorithm is not surprising.

It is not very useful to develop a general theory of divergence for fixed-point iteration, but we can clarify this by example. It is instructive to consider the simple case

$$g(x) := \alpha + \lambda(x - \alpha), \quad (2.10)$$

where for simplicity we take  $\lambda > 0$ . Then for all  $n$  we have

$$|x_n - \alpha| = |g(x_{n-1}) - \alpha| = \lambda|x_{n-1} - \alpha|, \quad (2.11)$$

and by induction

$$|x_n - \alpha| = \lambda^n|x_0 - \alpha|, \quad (2.12)$$

where  $x_0$  is our starting value. If  $\lambda < 1$  this converges, but if  $\lambda > 1$  this diverges.

The affine example (2.10) not only gives an example of divergence when  $|g'(\alpha)| > 1$ , it also suggests the asymptotic behavior of fixed-point iteration. When  $0 < |g'(\alpha)| < 1$ , the asymptotic behavior of fixed point iteration is (cf. Exercise 2.5) given by

$$|x_n - \alpha| \approx C|g'(\alpha)|^n \quad (2.13)$$

as  $n \rightarrow \infty$ , where  $C$  is a constant that depends on  $g$  and the initial guess.

### 2.1.2 Second-order iterations

What happens if  $g'(\alpha) = 0$ ? By Taylor's theorem

$$g(x) - \alpha = \frac{1}{2}(x - \alpha)^2 g''(\xi) \quad (2.14)$$

for some  $\xi$  between  $x$  and  $\alpha$ , and thus the error is squared at each iteration:

$$e^n = \frac{1}{2}(e^{n-1})^2 g''(\xi_n), \quad (2.15)$$

where  $\xi_n \rightarrow \alpha$  if the iteration converges. Of course, squaring the error is not a good thing if it is too large initially (with regard to the size of  $g''$ ).

We can now see why Heron's method converges so rapidly. Recall that  $g(x) = \frac{1}{2}(x + y/x)$ , so that  $g'(x) = \frac{1}{2}(1 - y/x^2) = 0$  when  $x^2 = y$ .

### 2.1.3 Higher-order iterations

It is possible to have even higher-order iterations. If  $g(\alpha) = \alpha$  and  $g'(\alpha) = g''(\alpha) = 0$ , then Taylor's theorem implies that

$$g(x) - \alpha = \mathcal{O}((x - \alpha)^3). \quad (2.16)$$

In principle, any order of convergence could be obtained [4]. However, while there is a qualitative change from geometric convergence to quadratic convergence, all higher order methods behave essentially the same. For example, given a second-order method, we can always create one that is fourth order just by taking two steps and calling them one. That is, we define

$$x_{\nu+1} = g(g(x_\nu)). \quad (2.17)$$

We could view this as introducing a 'half step'

$$x_{\nu+1/2} = g(x_\nu) \quad \text{and} \quad x_{\nu+1} = g(x_{\nu+1/2}). \quad (2.18)$$

Applying (2.15) twice, we see that  $x_{\nu+1} - \alpha = C(x_\nu - \alpha)^4$ . We can also verify this by defining  $G(x) = g(g(x))$  and evaluating derivatives of  $G$ :

$$\begin{aligned} G'(x) &= g'(g(x))g'(x) \\ G''(x) &= g''(g(x))g'(x)^2 + g'(g(x))g''(x) \\ G'''(x) &= g'''(g(x))g'(x)^3 + 3g''(g(x))g'(x)g''(x) + g'(g(x))g'''(x). \end{aligned} \quad (2.19)$$

Using the facts that  $g(\alpha) = \alpha$  and  $g'(\alpha) = 0$ , we see that  $G'(\alpha) = G''(\alpha) = G'''(\alpha) = 0$ . Thus, if  $\epsilon$  is the initial error, then the sequence of errors in a quadratic method (suitably scaled) is  $\epsilon, \epsilon^2, \epsilon^4, \epsilon^8, \epsilon^{16}, \dots$ , whereas for a fourth order method the sequence of errors (suitably scaled) is  $\epsilon, \epsilon^4, \epsilon^{16}, \dots$ . That is, the sequence of errors for the fourth-order method is just a simple subsequence (omit every other term) of the sequence of errors for the quadratic method.

What is not so clear at this point is that it is possible to have fractional orders of convergence. We will introduce a method with this property to illustrate how this is possible in Section 2.2.3.

## 2.2 Particular methods

Now we consider solving a general nonlinear equation of the form

$$f(\alpha) = 0. \quad (2.20)$$

Several methods use a geometric technique designed to point to a good place to look for the root:

$$x_{\nu+1} = x_\nu - \frac{f(x_\nu)}{s}, \quad (2.21)$$

where  $s$  is the slope of a line drawn from the point  $(x_\nu, f(x_\nu))$  to the next iterate  $x_{\nu+1}$ . This line is intended to intersect the  $x$ -axis at, or near, the root of  $f$ . This is based on the idea that the linear function  $\ell$  with slope  $s$  which is equal to  $f(x_\nu)$  at  $x_\nu$  vanishes at  $x_{\nu+1}$  (so  $\ell(x) = s(x_{\nu+1} - x)$ ).

The simplest fixed-point iteration might be to choose  $g(x) = x - f(x)$ . The geometric method can be viewed as a damped version of this, where we take instead  $g(x) = x - f(x)/s$ . You can think of  $s$  as an adjustment parameter to help with convergence of the standard fixed point iteration. The convergence of the geometric method is thus determined by the value of

$$g'(\alpha) = 1 - f'(\alpha)/s. \quad (2.22)$$

Ideally, we would simply pick  $s = f'(\alpha)$  if we knew how to compute it. We now consider various ways to approximate this value of  $s$ .

### 2.2.1 Newton's method

The method in question was a joint effort of both Newton<sup>1</sup> and his contemporary Joseph Raphson.<sup>2</sup> Both because Newton is better known, and because the full name is a bit long, we tend to abbreviate the name by dropping Raphson's name, but for now let us retain it.

The Newton-Raphson method chooses the slope adaptively at each stage:

$$s = f'(x_\nu). \quad (2.23)$$

The geometric method can be viewed as a type of difference approximation to this since we choose

$$s = \frac{0 - f(x_\nu)}{x_{\nu+1} - x_\nu}, \quad (2.24)$$

and we are making the approximation  $f(x_{\nu+1}) \approx 0$  in defining the difference quotient.

The Newton-Raphson method is sufficiently important that we should write out the iteration in detail:

$$x_{\nu+1} = x_\nu - \frac{f(x_\nu)}{f'(x_\nu)}. \quad (2.25)$$

Thus we see this as fixed point iteration with

$$g(x) = x - \frac{f(x)}{f'(x)}. \quad (2.26)$$

If  $x_\nu \rightarrow \alpha$ , then  $f'(x_\nu) \rightarrow f'(\alpha)$ , and so (2.22) should imply that the method is second-order convergent.

---

<sup>1</sup>Isaac Newton (1643–1727) is one of the best known scientists of all time [45].

<sup>2</sup>According to the mathematical historian Florian Cajori the approximate dates are 1648-1715.

Again, the second-order convergence of Newton's method is sufficiently important that it requires an independent computation:

$$\begin{aligned} g'(x) &= 1 - \frac{f'(x)^2 - f(x)f''(x)}{f'(x)^2} \\ &= \frac{f(x)f''(x)}{f'(x)^2}. \end{aligned} \quad (2.27)$$

Thus we conclude that  $f(\alpha) = 0$  implies  $g'(\alpha) = 0$ , provided that  $f'(\alpha) \neq 0$ . Thus Newton's method is second-order convergent provided  $f'(\alpha) \neq 0$  at the root  $\alpha$  of  $f(\alpha) = 0$ .

To estimate the convergence rate, we simply need to calculate  $g''$ :

$$\begin{aligned} g''(x) &= \frac{d}{dx} \left( \frac{f(x)f''(x)}{f'(x)^2} \right) \\ &= \frac{f'(x)^3 f''(x) + f(x)f'(x)^2 f^{(3)}(x) - 2f(x)f'(x)f''(x)^2}{f'(x)^4} \\ &= \frac{f''(x)}{f'(x)} + \frac{f(x)}{f'(x)^3} (f'(x)f^{(3)}(x) - 2f''(x)^2) \end{aligned} \quad (2.28)$$

Assuming  $f'(\alpha) \neq 0$ , this simplifies for  $x = \alpha$ :

$$g''(\alpha) = \frac{f''(\alpha)}{f'(\alpha)}. \quad (2.29)$$

From (2.15), we expect that Newton's method converges asymptotically like

$$e_{n+1} \approx \frac{1}{2} \frac{f''(\alpha)}{f'(\alpha)} e_n^2. \quad (2.30)$$

We can see that Heron's method is the same as Newton's method if we take  $f(x) = x^2 - y$ . We have

$$g(x) = x - \frac{f(x)}{f'(x)} = x - \frac{x^2 - y}{2x} = \frac{1}{2}x - \frac{y}{2x}. \quad (2.31)$$

Recall that  $g'(x) = \frac{1}{2}(1 - y/x^2)$ , so that  $g''(x) = y/x^3 = 1/\sqrt{y}$  when  $x = \sqrt{y}$ . Thus we can assert that Heron's method is precisely of second order.

## 2.2.2 Other second-order methods

The Steffensen iteration uses an adaptive difference:

$$s = \frac{f(x_\nu + f(x_\nu)) - f(x_\nu)}{f(x_\nu)} \quad (2.32)$$

in which the usual  $\Delta x = f(x_\nu)$  (which will go to zero: very clever). The iteration thus takes the form

$$x_{\nu+1} = x_\nu - \frac{f(x_\nu)^2}{f(x_\nu + f(x_\nu)) - f(x_\nu)} \quad (2.33)$$

We leave as Exercise 2.8 to verify that the iteration (2.33) is second-order convergent.

Steffensen's method is of the same order as Newton's method, but it has the advantage that it does not require evaluation of the derivative. If the derivative of  $f$  is hard to evaluate, this can be an advantage. On the other hand, it does require two function iterations each iteration, which could make it comparable to Newton's method, depending on whether it is easier or harder to evaluate  $f'$  versus  $f$ . Unfortunately, Steffensen's method does not generalize to higher dimensions, whereas Newton's method does (Section 8.2).

### 2.2.3 Secant method

The **secant method** approximates the slope by a difference method:

$$s = \frac{f(x_\nu) - f(x_{\nu-1})}{x_\nu - x_{\nu-1}}. \quad (2.34)$$

The error behavior is neither first nor second order, but rather something in between. Let us derive an expression for the sequence of errors.

First let us write out the method in the usual form:

$$x_{\nu+1} = x_\nu - \frac{(x_\nu - x_{\nu-1}) f(x_\nu)}{f(x_\nu) - f(x_{\nu-1})} \quad (2.35)$$

Subtracting  $\alpha$  from both sides, and inserting  $\alpha - \alpha$  in the numerator on the right-hand side and expanding, we find

$$\begin{aligned} e_{\nu+1} &= e_\nu - \frac{(e_\nu - e_{\nu-1}) f(x_\nu)}{f(x_\nu) - f(x_{\nu-1})} \\ &= \frac{-e_\nu f(x_{\nu-1}) + e_{\nu-1} f(x_\nu)}{f(x_\nu) - f(x_{\nu-1})} \\ &= \frac{x_\nu - x_{\nu-1}}{f(x_\nu) - f(x_{\nu-1})} \frac{-e_\nu f(x_{\nu-1}) + e_{\nu-1} f(x_\nu)}{x_\nu - x_{\nu-1}} \\ &= \frac{x_\nu - x_{\nu-1}}{f(x_\nu) - f(x_{\nu-1})} \frac{e_\nu (f(\alpha) - f(x_{\nu-1})) + e_{\nu-1} (f(x_\nu) - f(\alpha))}{x_\nu - x_{\nu-1}} \\ &= \frac{x_\nu - x_{\nu-1}}{f(x_\nu) - f(x_{\nu-1})} \frac{e_\nu e_{\nu-1}}{x_\nu - x_{\nu-1}} \left( \frac{f(\alpha) - f(x_{\nu-1})}{e_{\nu-1}} - \frac{f(\alpha) - f(x_\nu)}{e_\nu} \right). \end{aligned} \quad (2.36)$$

By Taylor's Theorem, we can estimate the expression

$$\frac{f(x_\nu) - f(x_{\nu-1})}{x_\nu - x_{\nu-1}} \approx f'(\alpha). \quad (2.37)$$

In Section 9.2.3, we will formally define this approximation as the first divided difference  $f[x_{\nu-1}, x_\nu]$ , and we will also identify the expression

$$\begin{aligned} & \frac{1}{x_\nu - x_{\nu-1}} \left( \frac{f(\alpha) - f(x_{\nu-1})}{e_{\nu-1}} - \frac{f(\alpha) - f(x_\nu)}{e_\nu} \right) \\ &= \frac{1}{x_\nu - x_{\nu-1}} \left( -\frac{f(\alpha) - f(x_{\nu-1})}{\alpha - x_{\nu-1}} + \frac{f(\alpha) - f(x_\nu)}{\alpha - x_\nu} \right) \end{aligned} \quad (2.38)$$

as the second divided difference  $f[\alpha, x_{\nu-1}, x_\nu] \approx \frac{1}{2}f''(\alpha)$  (cf. (9.25)). Rather than get involved in all the details, let us just use these approximation to see what is going on. Thus we find

$$e_{\nu+1} \approx \frac{1}{2} \frac{f''(\alpha)}{f'(\alpha)} e_\nu e_{\nu-1}. \quad (2.39)$$

Thus the error is quadratic in the previous errors, but it is not exactly the square of the previous error. Instead, it is a more complicated combination.

To understand how the error is behaving, define a scaled error by  $\epsilon_\nu = M|e_\nu|$  where  $M$  is an upper-bound for  $\frac{1}{2}f''/f'$  in an interval containing all of the iterates. Then analogous to (2.39), one can prove (see Exercise 2.12) that

$$|e_{\nu+1}| \leq M|e_\nu| |e_{\nu-1}|. \quad (2.40)$$

This means that

$$\epsilon_{\nu+1} \leq \epsilon_\nu \epsilon_{\nu-1} \quad (2.41)$$

for all  $\nu$ . If  $\delta = \max\{\epsilon_0, \epsilon_1\}$ , then (2.41) means that  $\epsilon_2 \leq \delta^2$ ,  $\epsilon_3 \leq \delta^3$ ,  $\epsilon_4 \leq \delta^5$ , and so forth. In general,  $\epsilon_\nu \leq \delta^{f_\nu}$  where  $f_\nu$  is the Fibonacci sequence defined by

$$f_{\nu+1} = f_\nu + f_{\nu-1}, \quad (2.42)$$

with  $f_0 = f_1 = 1$ . Quadratic convergence would mean that  $\epsilon_\nu \leq \delta^{2^\nu}$ , but the Fibonacci sequence grows more slowly than  $2^\nu$ . However, it is possible to determine the asymptotic growth exactly. In fact, we can write (Exercise 2.13)

$$f_{\nu-1} = \frac{1}{\sqrt{5}} (r_+^\nu - r_-^\nu), \quad r_\pm = \frac{1 \pm \sqrt{5}}{2} \approx \begin{cases} 1.6180 & (+) \\ -0.6180 & (-) \end{cases}. \quad (2.43)$$

Thus the errors for the secant method go to zero like (Exercise 2.14)

$$e_{\nu+1} \approx C e_\nu^{r_+}. \quad (2.44)$$

One iteration of secant only requires one function evaluation, so it can be more efficient than second-order methods. Two iterations of secant often requires work comparable to one iteration of Newton, and thus a method in which one iteration is two iterations of secant has a faster convergence rate, since  $2r_+ > 2$ .

## 2.3 Complex roots

Let us consider the situation when there are complex roots of equations. For example, what happens when we apply Heron's algorithm with  $y < 0$ ? Let us write  $y = -t$  ( $t > 0$ ) to clarify things, so that

$$x_{n+1} = \frac{1}{2}(x_n - t/x_n) \quad (2.45)$$

Unfortunately, for real values of  $x_0$ , the sequence generated by (2.45) does not converge to anything. On the other hand, if we take  $x_0 = i\rho$  where  $i = \sqrt{-1}$  and  $\rho$  is real, then

$$x_1 = \frac{1}{2}(x_0 - t/x_0) = \frac{1}{2}(i\rho - t/(i\rho)) = \frac{1}{2}i(\rho + t/\rho), \quad (2.46)$$

since  $1/i = -i$ . By induction, if  $x_n = i\rho_n$  where  $\rho_n$  is real, then  $x_{n+1} = i\rho_{n+1}$  where  $\rho_{n+1}$  is also real. More precisely,

$$x_{n+1} = \frac{1}{2}(x_n - t/x_n) = \frac{1}{2}(i\rho_n - t/(i\rho_n)) = \frac{1}{2}i(\rho_n + t/\rho_n), \quad (2.47)$$

so that

$$\rho_{n+1} = \frac{1}{2}(\rho_n + t/\rho_n). \quad (2.48)$$

We see that (2.48) is just the Heron iteration for approximating  $\rho = \sqrt{t}$ . Thus, convergence is assured as long as we start with a non-zero value for  $\rho$  (cf. Exercise 1.7).

The fact that Heron's method does not converge to an imaginary root given a real starting value is not spurious. Indeed, it is easy to see that Heron's method for a real root will also not converge if we start with a pure imaginary starting value. The set of values for which an iterative method converges is a valid study in dynamics, but here we will be mostly interested in the local behavior, that is, convergence given a suitable starting guess. It is not hard to see that reasonable methods converge when starting within some open neighborhood of the root.

For a general complex  $y$ , and a general starting guess  $x_0$ , it is not hard to see how Heron's algorithm will behave. Write  $z_n = x_n/\sqrt{y}$ . Then

$$z_{n+1} = x_{n+1}/\sqrt{y} = \frac{1}{2\sqrt{y}}(x_n + y/x_n) = \frac{1}{2}(z_n + 1/z_n). \quad (2.49)$$

Thus to study the behavior of Heron's method for complex roots, it suffices to study its behavior in approximating the square root of one with a complex initial guess (Exercise 2.10).

## 2.4 Error propagation

Suppose that the function  $g$  is not computed exactly. What can happen to the algorithm? Again, the affine function in (2.10) provides a good guide. Let us suppose that our computed function  $\hat{g}$  satisfies

$$\hat{g}(x) = g(x) + \delta(x) = \alpha + \lambda(x - \alpha) + \delta(x) \quad (2.50)$$

for some error function  $\delta(x)$ . If for example, we have  $\delta(x) = \delta > 0$  for all  $x$ , then  $\hat{g}(\hat{\alpha}) = \hat{\alpha}$  implies that

$$\hat{\alpha} = \hat{g}(\hat{\alpha}) = \alpha + \lambda(\hat{\alpha} - \alpha) + \delta = \alpha(1 - \lambda) + \lambda\hat{\alpha} + \delta \quad (2.51)$$

so that

$$\hat{\alpha} = \alpha + \frac{\delta}{1 - \lambda} \quad (2.52)$$

In [29], Theorem 3 on page 92 (see equation (18) there) provides a general theory, and (2.52) shows that the results there can be sharp.

The main problem with functional iteration in floating point is that the function may not really be continuous in floating point.

The effect of this kind of error on Newton's method is not so severe:

$$\hat{\alpha} - \alpha = \frac{\delta}{f'(\alpha)} \quad (2.53)$$

## 2.5 More reading

Techniques for iteratively computing functions have been essential for providing software (and firmware) to compute square roots, reciprocals and other basic mathematical operations. For further reading see the books [3] and [34].

The book [29] relates Steffensen's method to a general acceleration process due to Aitken<sup>3</sup> to accelerate convergence of sequences. The method of *false position*, or *regula falsi*, is a slight modification of the secant method in which  $x_{\nu-1}$  is replaced by  $x_k$ , where  $k$  is the last value where  $f(x_k)$  has a different sign from  $f(x_\nu)$  [17].

## 2.6 Exercises

**Exercise 2.1** *Suppose that  $g$  is a continuous function. Prove that, if fixed point iteration (2.3) converges to some  $\alpha$ , then  $\alpha$  is a fixed point, i.e., it satisfies (2.2).*

**Exercise 2.2** *Consider fixed point iteration to compute the solution of  $\cos \alpha = \alpha$ . Prove that this converges for any starting guess. Compute a few iterations to see what the approximate value of  $\alpha$  is.*

---

<sup>3</sup>Alexander Craig Aitken (1895–1967) was born in New Zealand and studied at the University of Edinburgh, where his thesis was considered so impressive that he was both appointed to a faculty position there and elected a fellow of the Royal Society of Edinburgh in 1925, before being awarded a DSc in 1926. He was elected to the Royal Society of London in 1936 for his work in statistics, algebra and numerical analysis. Aitken was reputedly one of the best mental calculators known, and had a prodigious memory [20, 28]. He was an accomplished writer, being elected to the Royal Society of Literature in 1964 in response to the publication of his war memoirs [1].

**Exercise 2.3** Suppose that  $g$  is a  $C^1$  function such that  $\alpha = g(\alpha)$  and with the property that  $|g'(\alpha)| < 1$ . Prove that fixed-point iteration (2.3) converges if  $x_0$  is sufficiently close to  $\alpha$ . (Hint: note by Lemma 2.1 that  $g$  is Lipschitz continuous with a constant  $\lambda < 1$  in an interval  $[\alpha - A, \alpha + A]$  for some  $A > 0$ .)

**Exercise 2.4** Suppose that  $g$  is a  $C^1$  function such that the Lipschitz estimate (2.4) holds on an interval  $[\alpha - A, \alpha + A]$  for some  $A > 0$ . Prove that  $|g'(\alpha)| \leq \lambda$ . (Hint: consider the difference quotients used to define  $g'(\alpha)$ .)

**Exercise 2.5** Suppose that  $g$  is a  $C^2$  function such that  $\alpha = g(\alpha)$  and with the property that  $|g'(\alpha)| < 1$ . Prove that the convergence of fixed-point iteration (2.3) converges asymptotically according to (2.13), that is,

$$\lim_{n \rightarrow \infty} \frac{|x_n - \alpha|}{|g'(\alpha)|^n} = C, \quad (2.54)$$

where  $C$  is a constant depending only on  $g$  and the initial guess  $x_0$ .

**Exercise 2.6** A function  $f$  is said to be Hölder continuous of exponent  $\alpha > 0$  provided that

$$|f(x) - f(y)| \leq \lambda|x - y|^\alpha \quad (2.55)$$

for all  $x$  and  $y$  in some interval. Show that the result (2.54) still holds as long as  $g'$  is Hölder continuous of exponent  $\alpha > 0$ .

**Exercise 2.7** Consider fixed point iteration  $x \leftarrow y/x$  for computing  $x = \sqrt{y}$ . Explain its behavior. Why does it not contradict the result in Exercise 2.3? (Hint: define  $g(x) = y/x$  and verify that a fixed point  $x = g(x)$  must satisfy  $x^2 = y$ . Perform a few iterations of  $x \leftarrow y/x$  and describe what you see. Evaluate  $g'$  at the fixed point.)

**Exercise 2.8** Prove that Steffensen's iteration (2.33) is second-order convergent provided that  $f'(\alpha) \neq 0$  at the root  $f(\alpha) = 0$  and  $f \in C^2$  near the root. (hint: write Steffensen's iteration as a fixed point iteration  $x_{\nu+1} = g(x_\nu)$  and show that  $g'(\alpha) = 0$  at the fixed point  $\alpha = g(\alpha)$ ).

**Exercise 2.9** Consider Newton's method for solving  $f(\alpha) = 0$  in the case that  $f'(\alpha) = 0$ . Show that second-order convergence is lost.

**Exercise 2.10** Investigate the behavior of the iteration (2.49) for various starting values of  $z_0$ . For what values of  $z_0$  does the iteration converge? For what values of  $z_0$  does the iteration not converge?

**Exercise 2.11** Investigate the fixed point(s) of the function

$$g(x) = \frac{1}{e^{-x} - 1} + \frac{1}{e^x - 1} + 1. \quad (2.56)$$

What is the value of  $g'$  at the fixed point(s)? (Hint: find a common denominator and simplify; see the solution to Exercise 12.10 for an application of this important function.)

**Exercise 2.12** Prove that (2.40) holds.

**Exercise 2.13** Prove that the Fibonacci numbers satisfy (2.43). (Hint: see Section 13.3.1; relate (2.42) with (13.15).)

**Exercise 2.14** Prove that the error in the secant method behaves as predicted in (2.44). (Hint: first prove (2.39).)

**Exercise 2.15** Develop algorithms to solve  $x^5 - x + a = 0$  (cf. (16.7)).

**Exercise 2.16** The expression  $f(x) = \sqrt{1+x} - 1$  arises in many computations. Unfortunately, for  $x$  small, round-off makes the obvious algorithm inaccurate; note that  $f(x) \approx \frac{1}{2}x$  for  $x$  small. Develop an algorithm to compute  $f$  that is accurate for  $|x| \leq \frac{1}{2}$ . (Hint: write  $t = \sqrt{1+x} - 1$  and observe that  $(1+t)^2 - 1 - x = 0$ . Try Newton's method starting with a good initial guess for  $t = f(x)$ .)

**Exercise 2.17** Consider the function

$$f(x) = \frac{1}{x} \left( \sqrt{x^2 + 1} - |x - 1| \right). \quad (2.57)$$

Develop an algorithm to compute  $f(x)$  with uniform accuracy for all  $0 < x < \infty$ . You may make reasonable assumptions about the accuracy of computing  $\sqrt{y}$ , but be explicit about them. (Hint: show that  $f(1/x) = xf(x)$  and that  $f(x) \approx 1 + \frac{1}{2}x + \mathcal{O}(x^3)$  for  $x$  small.)

**Exercise 2.18** In many applications in which roots  $f(x) = 0$  are sought, the cost of computing  $f$  (and  $f'$ ) is very large. You can simulate this via a loop

```
function y = f(x,n)
for i=1:n
    t = exp(x);
    y = log(t);
end
y=y*y-2;
```

This computes the function  $f(x) = x^2 - 2$ , but can take arbitrarily long to do so. Use an example like this to compare the efficiency of the Steffensen and secant methods for various values of  $n$ .

## 2.7 Solutions

**Solution of Exercise 2.2.** Define  $g(x) = \cos x$ . Then a solution to  $x = g(x)$  is what we are seeking, and we can apply fixed point iteration  $x_{n+1} = g(x_n)$ . This can be computed in `octave` or `Matlab` by simply repeating the command `x=cos(x)`, having started with, say,  $x = 1$ . After about forty iterations, it converges to  $\alpha = 0.73909$ . We have  $g'(\alpha) = -\sin \alpha \approx -0.67362$ . Thus it is clear that fixed point iteration is locally convergent.

The set of values  $x_n$  generated by fixed point iteration will always lie in  $[-1, 1]$ , the range of  $g(\cdot) = \cos \cdot$ . But we cannot easily assert global convergence, because the values of  $g'$  also extend over  $[-1, 1]$ . However,  $g$  maps  $[-1, 0]$  to  $[\beta, 1]$ , where  $\beta = \cos(-1) \approx 0.54030$ . Similarly,  $g$  maps  $[0, 1]$  to  $[\beta, 1]$ , but with the order reversed. Thus, regardless of the starting value  $x_0$ ,  $x_1 \in [\beta, 1]$ . Moreover, this argument shows further that all subsequent  $x_n \in [\beta, 1]$  as well. The maximum value of  $|g'(x)| = \sin x$  on the interval  $[\beta, 1]$  occurs at  $x = 1$ , since  $\sin$  is strictly increasing on  $[\beta, 1]$ , and  $\sin 1 = 0.84147$ . Thus we conclude that fixed point iteration converges at least as fast as  $0.84147^n$ .

**Solution of Exercise 2.5.** We have

$$x_{n+1} - \alpha = g(x_n) - g(\alpha) = g'(\xi_n)(x_n - \alpha) \quad (2.58)$$

for some  $\xi_n$  between  $x_n$  and  $\alpha$ . By induction, we thus find that

$$x_{n+1} - \alpha = \left( \prod_{i=0}^n g'(\xi_i) \right) (x_0 - \alpha). \quad (2.59)$$

Define  $r_i = |g'(\xi_i)/g'(\alpha)|$ . Then

$$\frac{|x_{n+1} - \alpha|}{|g'(\alpha)|^n} = \left( \prod_{i=0}^n r_i \right) |x_0 - \alpha|. \quad (2.60)$$

By (2.6), we know that  $|\xi_n - \alpha| \leq \lambda^n |e_0|$ . Therefore

$$|g'(\xi_n) - g'(\alpha)| = |g''(\hat{\xi}_n)(\xi_n - \alpha)| \leq |g''(\hat{\xi}_n)| \lambda^n |e_0|. \quad (2.61)$$

for some  $\hat{\xi}_n$  between  $\xi_n$  and  $\alpha$ , and therefore between  $x_n$  and  $\alpha$ . Therefore  $|r_n - 1| \leq \hat{C} \lambda^n$  where  $\hat{C}$  is chosen to be larger than  $|e_0 g''(\hat{\xi}_n)/g'(\alpha)|$  for all  $n$ . Since  $r_n \rightarrow 1$  as  $n \rightarrow \infty$ , we can be assured that  $r_n > 0$  for  $n$  sufficiently large. If  $r_n = 0$  for some value of  $n$ , then we have  $x_{n+i} = \alpha$  for all  $i \geq 1$ , so we can take the constant  $\hat{C}$  in (2.54) to be zero. If all,  $r_n > 0$ , then we can take the logarithm of the product in (2.60) to get

$$\log \left( \prod_{i=0}^n r_i \right) = \sum_{i=0}^n \log r_i. \quad (2.62)$$

Thus it suffices to prove that the sum on the right-hand side of (2.62) converges. Since we have an estimate on  $r_i - 1$ , we write  $r_i = 1 + \epsilon_i$ , where  $|\epsilon_i| \leq \hat{C}\lambda^i$ . Note that  $\log(1 + x) \leq x$  for all  $x > 0$ . To get such a bound for  $x < 0$ , set  $t = -x$  and write

$$|\log(1 - t)| = -\log(1 - t) = \int_{1-t}^1 \frac{dx}{x} \leq \frac{t}{1-t}. \quad (2.63)$$

Thus  $|\log(1 + x)| \leq 2|x|$  for  $|x| \leq \frac{1}{2}$ . Therefore,  $|\log r_i| \leq 2|1 - r_i| \leq \hat{C}\lambda^i$  for  $i$  sufficiently large, and thus the sum on the right-hand side of (2.62) to some value  $\gamma$ . Define  $C = e^\gamma|x_0 - \alpha|$ .

**Solution of Exercise 2.16.** We recall that

$$t = f(x) = \sqrt{1+x} - 1. \quad (2.64)$$

We seek an algorithm that outputs  $\hat{t}$  with the property that

$$|t - \hat{t}| = \left| \sqrt{1+x} - 1 - \hat{t} \right| \leq \frac{1}{2}\epsilon|t|, \quad (2.65)$$

where  $\epsilon > 0$  is a prescribed accuracy that we will require to be sufficiently small. To begin with, we establish some inequalities of use later. From (2.64), we can express

$$\frac{2|t|}{|x|} = \left| \frac{1}{x} \int_1^{1+x} y^{-3/2} dy \right| \quad (2.66)$$

as the (absolute value of the) mean value of the integrand on the right-hand side. Thus the quotient  $|t|/|x|$  is decreasing and its maximum for  $|x| \leq \frac{1}{2}$  occurs at  $x = -\frac{1}{2}$ . Thus (2.66) implies that

$$|t| \leq 0.6|x| \quad (2.67)$$

for  $|x| \leq \frac{1}{2}$ . For  $|x| \leq \frac{1}{2}$ , we also have

$$|t| \leq 0.3, \quad (2.68)$$

since  $t$  is an increasing function of  $x$  and the extreme value occurs again for  $x = -\frac{1}{2}$ . Adding one to both sides of (2.64) and squaring, we find that  $(t+1)^2 = 1+x$ , and thus

$$x = (t+1)^2 - 1 = 2t + t^2 = t(2+t). \quad (2.69)$$

Therefore, (2.68) implies that

$$|x| \leq 2.3|t|. \quad (2.70)$$

Now let us define  $\hat{t}$ . Taylor's theorem shows that

$$\left| \sqrt{1+x} - 1 - \frac{1}{2}x + \frac{1}{8}x^2 \right| \leq 0.1|x^3| \quad (2.71)$$

for  $x \in [-0.1, 0.1]$ . This means that for  $|x| \leq \sqrt{\epsilon}$  and  $\epsilon \leq 1/100$ , we can simply define

$$\hat{t} = \frac{1}{2}x - \frac{1}{8}x^2. \quad (2.72)$$

Then (2.71) implies that

$$\left| \sqrt{1+x} - 1 - \hat{t} \right| \leq 0.1|x|\epsilon \leq \frac{1}{4}|t|\epsilon, \quad (2.73)$$

by (2.70). This proves (2.65) when  $|x| \leq \sqrt{\epsilon}$ .

Thus we can turn our attention to the case where  $|x| > \sqrt{\epsilon}$ . In view of (2.69), the function  $\phi(\tau) = (1 + \tau)^2 - 1 - x = 2\tau + \tau^2 - x = 0$  when  $\tau = t$ . Thus  $t = f(x)$  is the solution to  $\phi(t) = 0$ , and we can consider using Newton's method to find  $t$ . Differentiating, we have  $\phi'(\tau) = 2 + 2\tau$ . Thus Newton's method is

$$t_{\nu+1} = t_{\nu} - \frac{2t_{\nu} + t_{\nu}^2 - x}{2 + 2t_{\nu}} = \frac{t_{\nu}^2 + x}{2 + 2t_{\nu}} =: g(t_{\nu}). \quad (2.74)$$

Differentiating again, we find that

$$g'(\tau) = \frac{2\tau + \tau^2 - x}{2(1 + \tau)^2} = \frac{\phi(\tau)}{2(1 + \tau)^2} = \frac{1}{2} - \frac{x + 1}{2(1 + \tau)^2}. \quad (2.75)$$

Therefore  $\phi(t) = 0$  implies  $g'(t) = 0$ , and differentiating yet again, we have

$$g^{(2)}(\tau) = -(x + 1)(1 + \tau)^{-3}. \quad (2.76)$$

Now let us consider using  $t_0 = f\ell\sqrt{f\ell(1+x)} - 1$  as starting guess. More precisely, we define  $a = f\ell(1+x)$ ,  $b = f\ell\sqrt{a}$ , and  $t_0 = f\ell(b-1)$ . We assume an estimate like (2.65) holds for all floating point operations, namely we assume that  $a = (1+x)(1+\delta_1)$ ,  $b = \sqrt{a}(1+\delta_2)$ , and  $t_0 = (b-1)(1+\delta_3)$  where  $|\delta_i| \leq \epsilon$ . For simplicity, we write  $1 + \delta_1 = (1 + \hat{\delta}_1)^2$ ; that is,  $\hat{\delta}_1 = \sqrt{1 + \delta_1} - 1 = f(\delta) \approx \frac{1}{2}\delta_1$ . In particular,  $|\hat{\delta}_1| \leq 0.6\delta_1$  as long as  $\epsilon \leq \frac{1}{2}$ , by (2.67).

Therefore  $\sqrt{a} = \sqrt{1+x}(1 + \hat{\delta}_1)$ , and

$$b = \sqrt{1+x}(1 + \hat{\delta}_1)(1 + \delta_2) = \sqrt{1+x} \left( 1 + \hat{\delta}_1 + \delta_2 + \hat{\delta}_1\delta_2 \right). \quad (2.77)$$

Since  $t = \sqrt{1+x} - 1$ , we find

$$b - 1 - t = \sqrt{1+x} \left( \hat{\delta}_1 + \delta_2 + \hat{\delta}_1\delta_2 \right). \quad (2.78)$$

Recall that  $t_0 = (b-1)(1+\delta_3)$ , so that  $t_0 - (b-1) = (b-1)\delta_3$ , and thus

$$\begin{aligned} t_0 - t &= t_0 - (b-1) + (b-1) - t \\ &= (b-1)\delta_3 + \sqrt{1+x} \left( \hat{\delta}_1 + \delta_2 + \hat{\delta}_1\delta_2 \right) \\ &= t\delta_3 + \sqrt{1+x} \left( \hat{\delta}_1 + \delta_2 + \hat{\delta}_1\delta_2 \right) (1 + \delta_3) \\ &= t\delta_3 + (t+1) \left( \hat{\delta}_1 + \delta_2 + \hat{\delta}_1\delta_2 \right) (1 + \delta_3). \end{aligned} \quad (2.79)$$

In particular, (2.79) shows why  $t_0$  cannot be used as a uniform approximation to  $t$ , since the second term is multiplied by  $t + 1$  and not  $t$ . However, if  $\epsilon \leq 10^{-2}$  (as we assumed above), then (2.79) implies that  $|t_0 - t| \leq 2.41\epsilon$ . If we define  $t_1$  by taking one Newton step (2.74), then (2.14) and (2.76) imply that

$$|t_1 - t| \leq 3\epsilon^2(x + 1)(1 + \xi)^{-3} = 3\epsilon^2(t + 1)^2(1 + \xi)^{-3}, \quad (2.80)$$

where  $\xi$  lies between  $t_0$  and  $t$ . By (2.68),  $t \geq -0.3$ , and certainly  $t_0 > -0.31$ . Therefore, a very crude estimate give

$$|t_1 - t| \leq 16\epsilon^2 \leq 16\epsilon^{3/2}|x| \leq 37\epsilon^{3/2}|t|, \quad (2.81)$$

by (2.70). This proves (2.65) provided  $\epsilon \leq 1/5476$ , as we now require.