Part II

Sequences of Numbers

CHAPTER 6

Difference Equations and Round-off Errors

An important ingredient in school mathematics is solution of algebraic equations like $x+3=4$. The challenge is to determine a numerical value for x such that the equation holds. In this chapter we are going to give a brief review of *difference equations* or *recurrence relations*. In contrast to traditional equations, the unknown in a difference equation is not a single number, but a sequence of numbers.

For some simple difference equations, an explicit formula for the solution can be found with pencil-and-paper methods, and we will review some of these methods in section 6.4. For most difference equations, there are no explicit solutions. However, a large group of equations can be solved numerically on a computer, and in section 6.3 we will see how this can be done.

In chapter 5 we saw real numbers are approximated by floating-point numbers, and how the limitations inherent in floatng-point numbers sometimes may cause dramatic errors. In section 6.5 we will see how round-off errors affect the numerical solutions of difference equations.

6.1 Why equations?

The reason equations are so useful is that they allow us to characterise unknown quantites in terms of natural principles that may be formulated as equations. Once an equation has been written down, we can apply standard techniques for solving the equation and determining the unknown. To illustrate, let us consider a simple example.

A common type of puzzle goes like this: *Suppose a man has a son that is half* *his age, and the son will be 16 years younger than his father in 5 years time. How old are they?*

With equations we do not worry about the ages, but rather write down what we know. If the age of the father is x and the age of the son is y, the first piece of information can be expressed as $y = x/2$, and the second as $y = x - 16$. This has given us two equations in the two unknowns x and y,

$$\begin{aligned} y &= x/2, \\ y &= x - 16. \end{aligned}$$

Once we have the equations we use standard techniques to solve them. If this case, we find that $x = 32$ and $y = 16$. This means that the father is 32 years old, and the son 16.

Difference equations work in the same way, except that they allow us to model phenomena where the unknown is a sequence of values, like the annual growth of money in a bank account, or the size of a population of animals over a period of time. The difference equation is obtained from known principles, and then the equation is solved by a mathematical or numerical method.

6.2 Difference equations defined

A simple difference equation arises if we try to model the growth of money in a bank account. Suppose that the amount of money in the account after n years is x_n, and the interest rate is 5 % per year. If interest is added once a year, the amount of money after $n+1$ years is given by the difference equation

$$x_{n+1} = x_n + 0.05x_n = 1.05x_n. \tag{6.1}$$

This equation characterises the growth of all bank accounts with a 5 % interest rate — in order to characterise a specific account we need to know how much money there was in the account to start with. If the initial deposit was 100000 (in your favourite currency) at time $n = 0$, we have an *initial condition* $x_0 = 100000$. This gives the complete model

$$x_{n+1} = 1.05x_n, \quad x_0 = 100000. \tag{6.2}$$

This is an example of a *first-order* difference equation with an initial condition. Suppose that we withdraw 1000 from the account every year. If we include this in our model we obtain the equation

$$x_{n+1} = 1.05x_n - 1000. \tag{6.3}$$

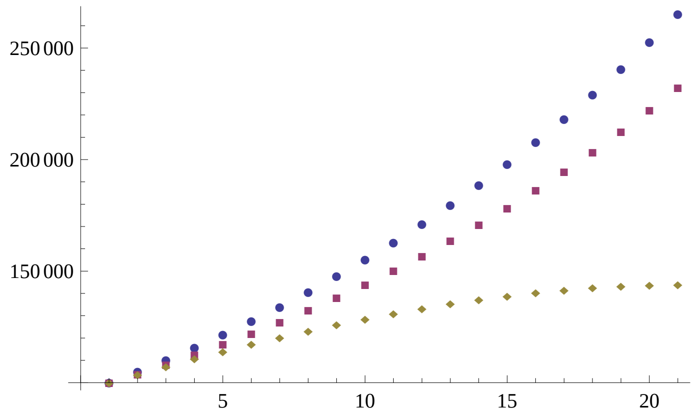

Figure 6.1. The growth of capital according to the models (6.1) (largest growth), (6.3) (middle growth), and (6.4) (smallest growth).

As the capital accumulates, it is reasonable that the owner increases the withdrawals. If for example the amount withdrawn increases by 300 each year, we get the model

$$x_{n+1} = 1.05x_n - (1000 + 300n). \tag{6.4}$$

A plot of the development of the capital in the three different cases is shown in figure 6.1. Note that in the case of (6.4) it appears that the growth of the capital levels out. In fact, it can be shown that after about 45 years, all the capital will be lost.

After these simple examples, let us define difference equations in general.

Definition 6.1 (Difference equation)**.** *A* difference equation *or* recurrence relation *is an equation that involves the terms of an unknown sequence* (x_n)*. The equation is said to be of order k if a term in the sequence depends on k previous terms, as in*

$$x_{n+k} = f(n, x_n, x_{n+1}, \dots, x_{n+k-1}), \tag{6.5}$$

where f is a function of $k+1$ variables. The actual values of n for which (6.5) *should hold may vary, but would typically be all nonzero integers.*

It is instructive to see how the three examples above fit into the general setting of definition 6.1. In all three cases we have $k = 1$; in the case (6.1) we have $f(t,x) = 0.5x$, in (6.3) we see that$f(t,x) = 0.5x - 1000$, and in (6.4) we have $f(t,x) = 1.05x - (1000 + 300t)$.

The examples above all led to a simple first-order difference equation. Here is an example where we end up with an equation of higher order.

Example 6.2. Suppose we have a population of pairs of animals where all pairs have one pair of babies, and the gestation period is three months. After having given birth, an animal can become pregnant again straightaway. This means that in month n, the population consists of all the pairs from the previous month. In addition, we have the babies born this month; these are the babies that were conceived three months ago, when the population was x_{n-3}. This gives the total model

$$x_n = x_{n-1} + x_{n-3}, \quad n = 3, 4, \dots,$$

or

$$x_{n+3} = x_{n+2} + x_n, \quad n = 0, 1, \dots$$

We obtain a difference equation of order k if we assume that the animals have a gestation period of k months. By reasoning in the same way we then find

$$x_{n+k} = x_{n+k-1} + x_n, \quad n = 0, 1, \dots \tag{6.6}$$

In the case $k = 2$ we get the famous Fibonacci model. ∎

Difference equations are particularly nice from a computational point of view since we have an explicit formula for a term in the sequence in terms of previous terms. In the bank example above, next year's balance is given explicitly in terms of this year's balance in formulas (6.1), (6.3), and (6.4). For general equations, we can compute x_{n+k} from the k previous terms in the sequence, as in (6.5). In order for this to work, we must be able to start somewhere, i.e., we need to know k consecutive terms in the sequence. It is common to assume that these terms are $x_0, \dots, x_{k-1}$, but they could really be any k consecutive terms.

Observation 6.3 (Initial conditions). *For a difference equation of order k, the solution is uniquely determined if k consecutive values of the solution is specified. These* initial conditions *are usually given as*

$$x_0 = a_0, \quad x_1 = a_1, \quad \dots \quad x_k = a_k,$$

where $a_0, \dots, a_k$ are given numbers.

Note that the number of initial conditions required equals the order of the equation. The model for population growth (6.6) therefore requires k initial conditions. A natural way to choose the initial conditions in this model is to set

$$x_0 = \cdots = x_k = 1. \tag{6.7}$$

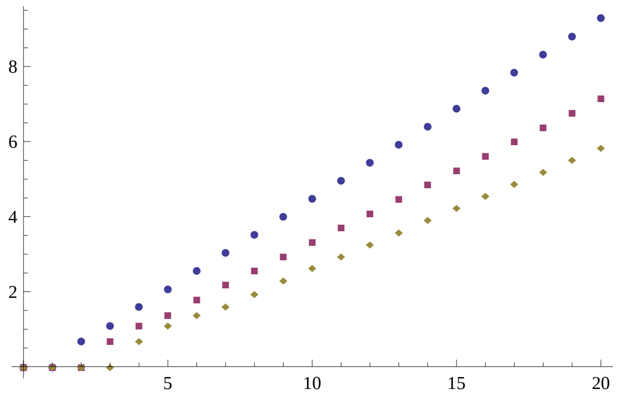

Figure 6.2. Plots of the natural logarithm of the solution of the difference equation (6.6) for $k = 2$ (fastest growth), 3 (medium growth), and 4 (slowest growth).

This corresponds to starting with a population of one new-born pair which remains the only one until this pair gives birth to another pair after k months. Plots of the logarithm of the solutions for $k = 2$, 3 and 4 are shown in figure 6.2. Since the logarithm is a straight line,

$$\ln x_n = a + bn,$$

where a and b are constants that can be found from the graphs, we have

$$x_n = e^{a+bn} = Ce^{bn},$$

with $C = e^a$. In other words, the solutions of (6.6), with initial values given by (6.7), grow exponentially.

We end this section by introducing some more terminology.

Definition 6.4. *A kth-order difference equation is said to be* linear *and* inhomogenous *if it has the form*

$$x_{n+k} = g(n) + f_0(n)x_n + f_1(n)x_{n+1} + \cdots + f_{k-1}(n)x_{n+k-1},$$

where g and $f_0, \ldots, f_{k-1}$ are functions of n. It is said to have constant coefficients *if the functions $f_0 \ldots, f_{k-1}$ do not depend on n. It is said to be homogenous if $g(n) = 0$ for all n.*

From this definition we see that the all the difference equations we have encountered so far have been linear, with constant coefficients. The equations (6.3) and (6.4) are inhomogenous, the others are homogenous.

6.3 Simulating difference equations

We mentioned above that difference equations are well suited for computations. More specifically, it is easy to calculate the solution numerically since we have an explicit formula for the next term in the sequence. Let us start by doing this for second-order linear equations. This is an equation in the form

$$x_{n+2} = g(n) + f_0(n)x_n + f_1(n)x_{n+1}, \quad x_0 = a_0,\, x_1 = a_1 \tag{6.8}$$

where g, f_0 and f_1 are given functions of n, and a_0 and a_1 are given real numbers. The following is an algorithm for computing the first terms of the solution $\{x_n\}$.

Algorithm 6.5. *Suppose the second-order equation* (6.8) *is given, i.e., the functions g, f_0, and f_1 are given together with the initial values a_0 and a_1. The following algorithm will compute the first $N+1$ terms x_0, x_1, …, x_N of the solution:*

```
$x_0 := a_0$;
$x_1 := a_1$;
**for** $i := 2, 3, \dots, N$
    $x_i := g(i) + f_0(i)x_{i-2} + f_1(i)x_{i-1}$;
```

This algorithm computes all the $N+1$ terms and saves them in the array $\boldsymbol{x} = [x_0, \dots, x_N]$. Sometimes we are only interested in the last term x_N, or we just want to print out the terms as they are computed — then there is no need to store all the terms.

Algorithm 6.6. *The following algorithm computes the solution of* (6.8)*, just like algorithm 6.5, but prints each term instead of storing them:*

```
$x_{pp} := a_0$;
$x_p := a_1$;
**for** $i := 2, 3, \dots, N$
    $x := g(i-2) + f_0(i-2)x_{pp} + f_1(i-2)x_p$;
    **print** $x$;
    $x_{pp} := x_p$;
    $x_p := x$;
```

The algorithm is based on the simple fact that in order to compute the next term, we only need to know the two previous terms, since the equation is of second order. At time i, the previous term x_{i-1} is stored in x_p and the term x_{i-2}

is stored in x_{pp}. Once x_i has been computed, we must prepare for the next step and make sure that x_p is shifted down to x_{pp}, which is not needed anymore, and x is stored in x_p. Note that it is important that these assignments are performed in the right order. At the beginning, the values of x_p and x_{pp} are given by the initial values.

In both of these algorithms it is assumed that the coefficients given by the functions g, f_0 and f_1, as well as the initial values a_0 and a_1, are known. In practice, the coefficient functions would usually be entered as functions (or methods) in the programming language you are using, while the initial values could be read from the terminal or via a graphical user interface.

Algorithms (6.5) and (6.6) can easily be generalised to 3rd or 4th order, or equations of any fixed order. The most convenient is to have an algorithm which takes the order of the equation as input.

Algorithm 6.7. *The following algorithm computes and prints the first $N+1$ terms of the solution of the kth-order difference equation*

$$x_{n+k} = f(n, x_n, x_{n+1}, \dots, x_{n+k-1}), \quad n = 0, 1, \dots, N-k \tag{6.9}$$

with initial values $x_0 = a_0$, $x_1 = a_1$, …, $x_{k-1} = a_{k-1}$. Here f is a given function of $k+1$ variables, and a_0, …, a_{k-1} are given real numbers:

```
for $i := 0, 1, \dots, k-1$
    $z_i := a_i$;
    print $z_i$;
for $i := k, k+1, \dots, N$
    $x := f(i-k, z_0, \dots, z_{k-1})$;
    print $x$;
    for $j := 0, \dots, k-2$
        $z_i := z_{i+1}$;
    $z_{k-1} := x$;
```

Algorithm 6.7 is similar to algorithm 6.6 in that it does not store all the terms of the solution sequence. To compensate it keeps track of the k previous terms in the array $\boldsymbol{z} = [z_0, \dots, z_{k-1}]$. The values x_k, x_{k+1}, …, x_N are computed in the second **for**-loop. By comparison with (6.9) we observe that $i = n+k$; this explains $i - k = n$ as the first argument to f. The initial values are clearly correct the first time through the loop, and at the end of the loop they are shifted along so that the value in z_0 is lost and the new value x is stored in z_{k-1}.

Difference equations have the nice feature that a term in the unknown sequence is defined explicitly as a function of previous values. This is what makes it simple to generate the values by algorithms like the ones sketched here. Provided the algorithms are correct and all operations are performed without errors, the exact solution will be computed. When the algorithms are implemented on a computer, this is the case if all the initial values are integers and all computations can be performed without introducing fractional numbers. One example is the Fibonacci equation

$$x_{n+2} = x_n + x_{n+1}, \quad x_0 = 1,\, x_1 = 1.$$

However, if floating-point numbers are needed for the computations, round-off errors are bound to occur and it is important to understand how this affects the computed solution. This is quite difficult to analyse in general, so we will restrict our attention to linear equations with constant coefficients. First we need to review the basic theory of linear difference equations.

6.4 Review of the theory for linear equations

Linear difference equations with constant coefficients have the form

$$x_{n+k} + b_{k-1}x_{n+k-1} + \cdots + b_1 x_{n+1} + b_0 x_n = g(n)$$

where $b_0, \dots, b_{k-1}$ are real numbers and $g(n)$ is a function of one variable. Initially we will focus on first-order ($k = 1$) and second-order ($k = 2$) equations for which $g(n) = 0$ for all n (homogenous equations).

One may wonder why we need to analyse these equations in detail when we can simulate theme so easily numerically. One reason is that for degree 1 and 2, the analytic solutions are so simple that they have practical importance. For our purposes however, it is equally important that the theory to be presented below will help us to understand how round-off errors influence the results of numerical simulations — this is the main topic in section 6.5.

6.4.1 First-order homogenous equations

The general first-order linear equation with constant coefficients has the form

$$x_{n+1} = bx_n, \tag{6.10}$$

where b is some real number. Often we are interested in x_n for all $n \geq 0$, but any value of $n \in \mathbb{Z}$ makes sense in the following. From (6.10) we find

$$x_{n+1} = bx_n = b^2 x_{n-1} = b^3 x_{n-2} = \cdots = b^{n+1} x_0. \tag{6.11}$$

This is the content of the first lemma.

Lemma 6.8. *The first-order homogenous difference equation*

$$x_{n+1} = bx_n, \quad n \in \mathbb{Z},$$

where b is an arbitrary real number, has the general solution

$$x_n = b^n x_0, \quad n \in \mathbb{Z}.$$

If x_0 is specified, the solution is uniquely determined.

The fact that the solution also works for negative values of n follows just like in (6.11) if we rewrite the equation as $x_n = b^{-1}x_{n+1}$ (assuming $b \neq 0$).

We are primarily interested in the case where $n \geq 0$, and then we have the following simple corollary.

Corollary 6.9. *For $n \geq 0$, the solution of the difference equation $x_{n+1} = bx_n$ will behave according to one of the following four cases:*

$$\lim_{n\to\infty} x_n = \begin{cases} 0, & \text{if } |b| < 1; \\ \infty, & \text{if } |b| > 1; \\ x_0, & \text{if } b = 1; \\ (-1)^n x_0, & \text{if } b = -1. \end{cases}$$

Phrased differently, the solution of the difference equation will either tend to 0 or ∞, except in the case where $|b| = 1$.

6.4.2 Second-order homogenous equations

The general second-order homogenous equation is

$$x_{n+2} + b_1 x_{n+1} + b_0 x_n = 0. \tag{6.12}$$

The basic idea behind solving this equation is to try with a solution $x_n = r^n$ in the same form as the solution of first-order equations, and see if there are any values of r for which this works. If we insert $x_n = r^n$ in (6.12) we obtain

$$0 = x_{n+2} + b_1 x_{n+1} + b_0 x_n = r^{n+2} + b_1 r^{n+1} + b_0 r^n = r^n(r^2 + b_1 r + b_0).$$

In other words, we must either have $r = 0$, which is uninteresting, or r must be a solution of the quadratic equation

$$r^2 + b_1 r + b_0 = 0$$

which is called the *characteristic equation* of the difference equation. If the characteristic equation has the two solutions r_1 and r_2, we know that both $y_n = r_1^n$ and $z_n = r_2^n$ will be solutions of (6.12). And since the equation is linear, it can be shown that any combination

$$x_n = Cr_1^n + Dr_2^n$$

is also a solution of (6.12) for any choice of the numbers C and D. However, in the case that $r_1 = r_2$ this does not give the complete solution, and if the two solutions are complex conjugates of each other, the solution may be expressed in a more adequate form that does not involve complex numbers. In either case, the two free coefficients can be adapted to two initial conditions like $x_0 = a_0$ and $x_1 = a_1$.

Theorem 6.10. *The solution of the homogenous, second-order difference equation*

$$x_{n+2} + b_1 x_{n+1} + b_0 x_n = 0 \tag{6.13}$$

is governed by the solutions r_1 and r_2 of the characteristic equation

$$r^2 + b_1 r + b_0 = 0$$

as follows:

1. *If the two roots r_1 and r_2 are real and distinct, the general solution of* (6.13) *is given by*
$$x_n = Cr_1^n + Dr_2^n.$$
2. *If the two roots are equal, $r_1 = r_2$, the general solution of* (6.13) *is given by*
$$x_n = (C + Dn)r_1^n.$$
3. *If the two roots are complex conjugates of each other so that $r_1 = r$ and $r_2 = \bar{r}$, and r can be written in polar form as $r = \rho e^{i\theta}$, then the general solution of* (6.13) *is given by*
$$x_n = \rho^n (C \cos n\theta + D \sin n\theta).$$

In all three cases the solution can be determined uniquely by two initial conditions $x_0 = a_0$ and $x_1 = a_1$, where a_0 and a_1 are given real numbers, since this determines the two free coefficients C and D uniquely.

The proof of this theorem is not so complicated and can be found in a text on difference equations. The following is a consequence of the theorem which is analogous to corollary 6.9.

Corollary 6.11. *Suppose that one root, say r_1, of the characteristic equation satisfies $|r_1| > 1$, that $C \neq 0$, and that $|r_2| < |r_1|$. Then*

$$\lim_{n\to\infty} |x_n| = \infty.$$

On the other hand, if both $|r_1| < 1$ and $|r_2| < 1$, then

$$\lim_{n\to\infty} x_n = 0.$$

Note that in cases 2 and 3 in theorem 6.10, the two roots have the same absolute value (in case 2 the roots are equal and in case 3 they both have absolute value ρ). In those cases both roots will therefore either be larger than 1 in absolute value or smaller than 1 in absolute value. This means that it is only in the first case that we need to distinguish between the two roots in the conditions in corollary 6.11.

Proof of corollary 6.11. In cases 2 and 3 in theorem 6.10 $|r_1| = |r_1|$, so if $|r_1| > 1$ and $|r_2| < |r_1|$ we must have two real roots. Then we can write the solution as

$$x_n = r_1^n \left(C + D\left(\frac{r_2}{r_1}\right)^n \right)$$

and therefore

$$\lim_{n\to\infty} |x_n| = \lim_{n\to\infty} |r_1|^n \left| C + D\left(\frac{r_2}{r_1}\right)^n \right| = |C| \lim_{n\to\infty} |r_1| = \infty.$$

If both $|r_1| < 1$ and $|r_2| < 1$ and both roots are real, the triangle inequality leads to

$$\lim_{n\to\infty} |x_n| \leq \lim_{n\to\infty} \left(|C||r_1|^n + |D||r_2|^n \right) = 0.$$

If $r_1 = r_2$, and $|r_1| < 1$ (case 2 in theorem 6.10), we have the same conclusion since $n|r_1|^n$ tends to 0 when n tends to ∞. Finally, in the case of complex conjugate roots of absolute value less than 1 we have $\rho < 1$, so the term ρ^n will ensure that $|x_n|$ tends to 0. ∎

A situation that is not covered by corollary 6.11 is the case where both roots are real, but of opposite sign, and larger than 1 in absolute value. In this case the solution will also tend to infinity in most cases, but not always. Consider for example the case where $x_n = 2^n + (-2)^n$. Then $x_{2n+1} = 0$ for all n while $\lim_{n\to\infty} x_{2n} = \infty$.

6.4.3 Linear homogenous equations of general order

Consider now a kth-order, homogenous, and linear difference equation with constant coefficients,

$$x_{n+k} + b_{k-1}x_{n+k-1} + \cdots + b_1 x_{n+1} + b_0 x_n = 0,$$

where all the coefficients $\{b_i\}$ are real numbers. It is quite easy to show that if we have k solutions $\{x_n^i\}_{i=1}^k$, then the combination

$$x_n = C_1 x_n^1 + C_2 x_n^2 + \cdots + C_k x_n^k \tag{6.14}$$

will also be a solution for any choice of the coefficients $\{C_i\}$. As we have already seen, an equation of order k can be adapted to k initial values.

To determine k solutions, we follow the same procedure as for second-order equations and try the solution $x_n = r^n$. We then find that r must solve the characteristic equation

$$r^k + b_{k-1}r^{k-1} + \cdots + b_1 r + b_0 = 0.$$

From the fundamental theorem of algebra we know that this equation has k distinct roots, and complex roots occur in conjugate pairs since the coefficients are real. A theorem similar to theorem 6.10 can therefore be proved.

Observation 6.12. *The general solution of the difference equation*

$$x_{n+k} + b_{k-1}x_{n+k-1} + \cdots + b_1 x_{n+1} + b_0 x_n = 0$$

is a combination of k terms

$$x_n = C_1 x_n^1 + C_2 x_n^2 + \cdots + C_k x_n^k$$

where each term $\{x_k^i\}$ is a solution of the difference equation. The solution $\{x_n^i\}_n$ is essentially on the form $x_n^i = r_i^n$ where r_i is the ith root of the characteristic equation

$$r^k + b_{k-1}r^{k-1} + \cdots + b_1 r + b_0 = 0.$$

Note the word 'essentially' in the last sentence: just like for quadratic equations we have to take special care when there are double roots (or roots of even higher multiplicity) or complex roots.

Closed formulas for the roots can be found when for quadratic, cubic and quartic equations, but the expressions even for cubic equations, can be rather complicated. For higher degree than two one therefore has to resort to numerical techniques, like the ones in chapter 10, for finding the roots.

There is also a close of analog of corollary 6.11 which shows that a solution will tend to zero if all roots have absolute value less than 1. And if there is a root with absolute value greater than 1, whose corresponding coefficient in (6.14) is nonzero, then the solution will grow beyond all bounds when n becomes large.

6.4.4 Inhomogenous equations

So far we have only discussed homogenous difference equations. For inhomogenous equations there is an important, but simple lemma, which can be found in the standard text books on difference equations.

Lemma 6.13. *Suppose that $\{x_n^p\}$ is a particular solution of the inhomogenous difference equation*

$$x_{n+k} + b_{k-1}x_{n+k-1} + \cdots + b_1 x_{n+1} + b_0 x_n = g(n). \tag{6.15}$$

Then all other solutions of the inhomogenous equation will have the form

$$x_n = x_n^p + x_n^h$$

where $\{x_n^h\}$ is some solution of the homogenous equation

$$x_{n+k} + b_{k-1}x_{n+k-1} + \cdots + b_1 x_{n+1} + b_0 x_n = 0.$$

More informally, lemma 6.13 means that we can find the general solution of (6.15) by just finding one solution, and then adding the general solution of the homogenous equation. The question is how to find one solution. The following observation is useful.

Observation 6.14. *One of the solutions of the inhomogenous equation*

$$x_{n+k} + b_{k-1}x_{n+k-1} + \cdots + b_1 x_{n+1} + b_0 x_n = g(n)$$

has the same form as $g(n)$.

Some examples will illustrate how this works.

Example 6.15. Consider the equation

$$x_{n+1} - 2x_n = 3. \tag{6.16}$$

Here the right-hand side is constant, so we try with the a particular solution $x_n^h = A$, where A is an unknown constant to be determined. If we insert this in the equation we find

$$A - 2A = 3,$$

so $A = -3$. This means that $x_n^p = -3$ is a solution of (6.16). Since the general solution of the homogenous equation $x_{n+1} - 2x_n = 0$ is $x_n^h = C2^n$, the general solution of (6.16) is

$$x_n = x_n^h + x_n^p = C2^n - 3. \quad \blacksquare$$

In general, when $g(n)$ is a polynomial in n of degree d, we try with a particular solution which is a general polynomial of degree d. When this is inserted in the equation, we obtain a relation between two polynomials that should hold for all values of n, and this requires corresponding coefficients to be equal. In this way we obtain a set of equations for the coefficients.

Example 6.16. In the third-order equation

$$x_{n+3} - 2x_{n+2} + 4x_{n+1} + x_n = n \tag{6.17}$$

the right-hand side is a polynomial of degree 1. We therefore try with a solution $x_n^p = A + Bn$ and insert this in the difference equation,

$$n = A + B(n+3) - 2(A + B(n+2)) + 4(A + B(n+1)) + A + Bn = (4A + 3B) + 4Bn.$$

The only way for the two sides to be equal for all values of n is if the constant terms and first degree terms on the two sides are equal,

$$\begin{aligned} 4A + 3B &= 0, \\ 4B &= 1. \end{aligned}$$

From these equations we find $B = 1/4$ and $A = -3/16$, so one solution of (6.17) is

$$x_n^p = \frac{n}{4} - \frac{3}{16}. \quad \blacksquare$$

There are situations where the technique above does not work because the trial polynomial solution is also a homogenous solution. In this case the degree of the polynomial must be increased. For more details we refer to a text book on difference equations.

Other types of right-hand sides can be treated similarly. One other type is given by functions like

$$g(n) = p(n)a^n,$$

where $p(n)$ is a polynomial in n and a is a real number. In this case, one tries with a solution $x_n^p = q(n)a^n$ where $q(n)$ is a general polynomial in n of the same degree as $p(n)$.

Example 6.17. Suppose we have the equation

$$x_{n+1} + 4x_n = n3^n. \tag{6.18}$$

The right hand side is a first-degree polynomial in n multiplied by 3^n, so we try with a particular solution in the form

$$x_n^p = (A + Bn)3^n.$$

When this is inserted in the difference equation we obtain

$$\begin{aligned} n3^n &= \big(A + B(n+1)\big)3^{n+1} + 4(A + Bn)3^n \\ &= 3^n\Big(3\big(A + B(n+1)\big) + 4A + 4Bn\Big) \\ &= 3^n(7A + B + 5Bn). \end{aligned}$$

Here we can cancel 3^n, which reduces the equation to an equality between two polynomials. If these are to agree for all values of n, the constant terms and the linear terms must agree,

$$\begin{aligned} 7A + B &= 0, \\ 5B &= 1. \end{aligned}$$

This system has the solution $B = 1/5$ and $A = -1/35$, so a particular solution of (6.18) is

$$x_n^p = \Big(\frac{1}{5}n - \frac{1}{35}\Big)3^n.$$

The homogenous equation $x_{n+1} - 4x_n = 0$ has the general solution $x_n^h = C4^n$ so according to lemma 6.13 the general solution of (6.18) is

$$x_n = x_n^h + x_n^p = C4^n + \Big(-\frac{1}{35} + \frac{1}{5}n\Big)3^n. \quad \blacksquare$$

6.5 Round-off errors and stability for linear equations

In chapter 5, we saw that computations on a computer often lead to errors, at least when we use floating-point numbers. Therefore, when the solution of a difference equation is computed via one of the algorithms in section 6.3, we must be prepared for errors. In this section we are going to study this in some detail. We will restrict our attention to linear difference equations with constant coefficients.

We first recall that integer arithmetic is always correct, except for the possibility of overflow, which is so dramatic that it is usually quite easy to detect. We therefore focus on the case where floating-point numbers must be used. Note that we use 64-bit floating-point numbers in all the examples in this chapter.

The effect of round-off errors become quite visible from a couple of examples.

Example 6.18. Consider the equation

$$x_{n+2} - \frac{2}{3}x_{n+1} - \frac{1}{3}x_n = 0, \quad x_0 = 1,\, x_1 = 0. \tag{6.19}$$

Since the two roots of the characteristic equation $r^2 - 2r/3 - 1/3 = 0$ are $r_1 = 1$ and $r_2 = -1/3$, the general solution of the difference equation is

$$x_n = C + D\Big(-\frac{1}{3}\Big)^n.$$

The initial conditions yield the equations

$$\begin{aligned} C + D &= 1,\\ C - D/3 &= 0, \end{aligned}$$

which has the solution $C = 1/4$ and $D = 3/4$. The solution of (6.19) is therefore

$$x_n = \frac{1}{4}\big(1 + (-1)^n 3^{1-n}\big).$$

We observe that x_n tends to $1/4$ as n tends to infinity.

Note that if we simulate the equation (6.19) on a computer, the next term is computed by the formula $x_{n+2} = (2x_{n+1} + x_n)/3$. The division by 3 means that floating-point numbers are required to calculate this expression. If we simulate the difference equation, we obtain the four approximate values

$$\begin{aligned} \tilde{x}_{10} &= 0.250012701316,\\ \tilde{x}_{15} &= 0.249999947731,\\ \tilde{x}_{20} &= 0.250000000215,\\ \tilde{x}_{30} &= 0.250000000000, \end{aligned}$$

which agree with the exact solution to 12 digits. In other words, numerical simulation in this case works very well, even if floating-point numbers are used in the calculations. ∎

Example 6.19. We consider the difference equation

$$x_{n+2} - \frac{19}{3}x_{n+1} + 2x_n = -10, \quad x_0 = 2,\ x_1 = 8/3. \tag{6.20}$$

The two roots of the characteristic equation are $r_1 = 1/3$ and $r_2 = 6$, so the general solution of the homogenous equation is

$$x_n^h = C3^{-n} + D6^n.$$

To find a particular solution we try a solution $x_n^p = A$ which has the same form as the right-hand side. We insert this in the difference equation and find $A = 3$, so the general solution is

$$x_n = x_n^h + x_n^p = 3 + C3^{-n} + D6^n. \tag{6.21}$$

If we enforce the initial conditions, we end up with the system of equations

$$\begin{aligned} 2 = x_0 &= 3 + C + D, \\ 8/3 = x_1 &= 3 + C/3 + 6D. \end{aligned} \tag{6.22}$$

This may be rewritten as

$$\begin{aligned} C + D &= -1, \\ C + 18D &= -1. \end{aligned} \tag{6.23}$$

which has the solution $C = -1$ and $D = 0$. The final solution is therefore

$$x_n = 3 - 3^{-n}, \tag{6.24}$$

which tends to 3 when n tends to infinity.

Let us simulate the equation on the computer. As in the previous example we have to divide by 3 so we have to use floating-point numbers. Some early terms in the computed sequence are

$$\begin{aligned} \tilde{x}_5 &= 2.99588477366, \\ \tilde{x}_{10} &= 2.99998306646, \\ \tilde{x}_{15} &= 3.00001192858, \end{aligned}$$

(throughout this section we will use $\tilde{x}_n$ to denote a computed version of x_n). These values appear to approach 3 as they should. However, some later values are

$$\begin{aligned} \tilde{x}_{20} &= 3.09329859009, \\ \tilde{x}_{30} &= 5641411.98633, \\ \tilde{x}_{40} &= 3.41114428655 \times 10^{14}, \end{aligned} \tag{6.25}$$

and at least the last two of these are obviously completely wrong. ∎

6.5.1 Explanation of example 6.19

Let us see what went wrong in example 6.19. First of all we note that the initial values are $x_0 = 2$ and $x_1 = 8/3$. The first of these will be represented exactly in a computer whether we use integers or floating-point numbers. The second definitely requires floating-point numbers. Note though that the fraction 8/3 cannot be represented exactly with a finite number of digits, and therefore there will inevitably be round-off error.

Observation 6.20. *The initial value $x_0 = 2$ is represented exactly by floating-point numbers, but the initial value* 8/3 *at x_1 becomes $\tilde{x}_1 = a_1$, where a_1 is the floating-point number closest to* 8/3.

The fact that the initial value at x_1 is not quite correct means that when the coefficients C and D in (6.21) are determined, the solutions are not exactly $C = 1$ and $D = 0$. If the initial values used in computations are $\tilde{x}_0 = 2$ and $\tilde{x}_1 = 8/3 + \delta$ where δ is a small number (in the range 10^{-18} to 10^{-15} for 64-bit floating-point numbers), we can determine the new values of C and D from equations like (6.22). If we solve these equations we find

$$C = -1 - \frac{3}{17}\delta, \quad D = \frac{3}{17}\delta. \tag{6.26}$$

This is summarised in the next observation.

Observation 6.21. *Because of round-off errors in the second initial value, the exact values of the coefficients C and D in* (6.21) *used in calculations are*

$$C = -1 + \epsilon_1, \quad D = \epsilon_2 \tag{6.27}$$

where ϵ_1 and ϵ_2 are small numbers. The solution used in the simulations is therefore

$$\tilde{x}_n = 3 - (1 - \epsilon_1)3^{-n} + \epsilon_2 6^n.$$

The actual computations are based on the formula

$$x_{n+2} = \frac{19}{3}x_{n+1} - 2x_n - 10$$

for computing the next term in the solution. Here as well there is division by 3, which inevitably brings with it further round-off errors each time a new term is computed. It turns out that these errors may also be roughly accounted for by assuming that the initial values are changed.

Observation 6.22. *The computed solution of the difference equation* (6.20)*, including all round-off errors, is essentially the solution of a problem where the initial values have been changed, i.e., the computed values are the solution of the problem*

$$x_{n+2} - \frac{19}{3}x_{n+1} + 2x_n = -10, \quad x_0 = 2 + \delta_1,\ x_1 = 8/3 + \delta_2, \tag{6.28}$$

where the actual values of δ_1 and δ_2 depend on the details of the computations.

Observation 6.22 means that we should be able to predict the effect of round-off errors by solving a problem with slightly different initial values. When we change the initial values, the coefficients C and D also change. When we only perturbed x_1 we obtained the solutions given by (6.26). With the initial values in (6.28) we find

$$C = -1 + \frac{18\delta_1 - 3\delta_2}{17}, \quad D = \frac{3\delta_2 - \delta_1}{17}.$$

These solutions may be written as in (6.27) if we set

$$\begin{aligned} \epsilon_1 &= (18\delta_1 - 3\delta_2)/17, \\ \epsilon_2 &= (3\delta_2 - \delta_1)/17. \end{aligned} \tag{6.29}$$

The conclusion is therefore as in observation 6.21.

Observation 6.23. *When the difference equation* (6.20) *is simulated numerically, the computed numbers are essentially given by the formula*

$$\tilde{x}_n = 3 - (1 - \epsilon_1)3^{-n} + \epsilon_2 6^n, \tag{6.30}$$

where ϵ_1 and ϵ_2 are small numbers.

From observation 6.23 it is easy to explain where the values in (6.25) came from. Because of round-off errors, the computed solution is given by (6.30), where ϵ_2 is a small nonzero number. Even if ϵ_2 is small, the product $\epsilon_2 6^n$ will eventually become large, since 6^n grows beyond all bounds when n becomes large. In fact we can use the values in (6.25) to estimate ϵ_2. For $n = 40$ we have $3^{-n} \approx 8.2 \times 10^{-20}$ and $6^n \approx 1.3 \times 10^{31}$. Since we have used 64-bit floating-point numbers, this means that only the last term in (6.30) is relevant (the other two terms affect the result in about the 30th digit and beyond. This means that we can find ϵ_2 from the relation

$$3.4 \times 10^{14} \approx \tilde{x}_{40} \approx \epsilon_2 6^{40} \approx \epsilon_2 1.3 \times 10^{31}.$$

From this we see that $\epsilon_2 \approx 2.6 \times 10^{-17}$. This is a reasonable value since we know from (6.29) that ϵ_2 is roughly as large as the round-off error in the initial values. With 64-bit floating-point numbers we have about 15–18 decimal digits, so a round-off error of about 10^{-17} is to be expected when the numbers are close to 1 as in this example.

Observation 6.24. *When ϵ_2 is nonzero in* (6.30)*, the last term $\epsilon_2 6^n$ will eventually dominate the computed solution of the difference equation completely, and the computations will eventually end in overflow.*

6.5.2 Round-off errors for linear equations of general order

It is important to realise that the problem with the computed solution in (6.25) becoming large is not really because of unfortunately large round-off errors; any round-off error at all would give the same kind of behaviour. The general problem is that we have a family of solutions given by

$$x_n = 3 + C3^{-n} + D6^n, \tag{6.31}$$

and different initial conditions pick out different solutions (different values of C and D) within this family. With floating-point numbers it is basically impossible to get D to be exactly 0, so the last term in (13.4) will always dominate the computed solution for large values of n and completely overwhelm the other two parts of the solution.

The difference equation in example 6.19 is not particularly demanding — we will get the same effect whenever we have a difference equation where at least one of the roots of the characteristic equation is larger than 1 in absolute value, but the exact solution remains bounded, or at least significantly smaller than the part of solutions corresponding to the largest root.

Observation 6.25. *Suppose the difference equation*

$$x_{n+k} + b_{n-k}x_{n-k} + \cdots + b_1 x_{n+1} + b_0 x_n = g(n)$$

is solved (simulated) numerically with floating-point numbers, and let r be the root of the characteristic equation,

$$r^k + b_{k-1}r^{k-1} + \cdots + b_1 r + b_0 = 0,$$

with largest absolute value. If the particular solution of the inhomogenous equation does not grow as fast as $|r|^n$ (in case $|r| > 1$), or decays faster than $|r|^n$ (in the case $|r| < 1$), then the computed solution will eventually be dominated by the solution corresponding to the root r, regardless of what the initial values are.

In example 6.19, the solution family has three components: the two solutions 6^n and 3^{-n} from the homogenous equation, and the constant solution 3 from the inhomogenous equation. When the solution we are after just involves 3^{-n} and 3 we get into trouble because we invariably also bring along 6^n because of round-off errors. On the other hand, if the exact initial values lead to a solution that includes 6^n, then we will not get problems with round-off: The coefficient multiplying 6^n will be accurate enough, and the other terms are too small to pollute the 6^n solution.

Example 6.26. We consider the third-order difference equation

$$x_{n+3} - \frac{16}{3}x_{n+2} + \frac{17}{3}x_{n+1} - \frac{4}{3}x_n = 10 \times 2^n, \quad x_0 = -2,\ x_1 = -\frac{17}{3},\ x_2 = -\frac{107}{9}.$$

The coefficients have been chosen so that the roots of the characteristic equation are $r_1 = 1/3$, $r_2 = 1$ and $r_3 = 4$. To find a particular solution we try with $x_n^p = A2^n$. If this is inserted in the equation we find $A = -3$, so the general solution is

$$x_n = -3 \times 2^n + B3^{-n} + C + D4^n. \tag{6.32}$$

The initial conditions force $B = 0$, $C = 1$ and $D = 0$, so the exact solution is

$$x_n = 1 - 3 \times 2^n. \tag{6.33}$$

The discussion above shows that this is bound to lead to problems. Because of round-off errors, the coefficients B and D will not be exactly 0 when the equation is simulated. Instead we will have

$$\tilde{x}_n = -3 \times 2^n + \epsilon_1 3^{-n} + (1 + \epsilon_2) + \epsilon_3 4^n$$

Even if ϵ_3 is small, the term $\epsilon_3 4^n$ will dominate when n becomes large. This is confirmed if we do the simulations. The computed value $\tilde{x}_{100}$ is approximately 4.5×10^{43}, while the exact value is -3.8×10^{30}. ∎

6.6 Summary

In this chapter we met the effect of round-off errors on realistic computations for the first time. We saw that innocent-looking computations like the simulation of the difference equation in example 6.19 led to serious problems with round-off errors. By making use of the theory behind difference equations we were able to understand why the simulations behave they way they do. From this insight we also realise that for this particular equation and initial values, the blow-up is unavailable, just like cancellation is unavoidable when we subtract two almost equal numbers. Such problems are usually referred to as being *badly conditioned.* On the other hand, a different choice of initial conditions may lead to calculations with no round-off problems; then the problem is said to be *well conditioned.*

Exercises

6.1 Which equations are linear?

a) $x_{n+2} + 3x_{n+1} - \sin(n)x_n = n!$.

b) $x_{n+3} - x_{n+1} + x_n^2 = 0$.

c) $x_{n+2} + x_{n+1}x_n = 0$.

d) $nx_{n+2} - x_{n+1}e^n + x_n = n^2$.

6.2 Program algorithm 6.6 and test it on the Fibonacci equation

$$x_{n+2} = x_{n+1} + x_n, \quad x_0 = 0,\, x_1 = 1.$$

6.3 Generalise algorithm 6.6 to third order equations and test it on the Fibonacci like equation

$$x_{n+3} = x_{n+2} + x_{n+1} + x_n, \quad x_0 = 0,\, x_1 = 1,\, x_2 = 1.$$

6.4 In this exercise we are going to study the difference equation

$$x_{n+1} - 3x_n = 5^{-n}, \quad x_0 = -5/14. \tag{6.34}$$

a) Show that the general solution of (6.34) is

$$x_n = C3^n - \frac{5}{14}5^{-n}$$

and that the initial condition leads to the solution

$$x_n = -\frac{5}{14}5^{-n}.$$

b) Explain what will happen if you simulate equation 6.34 numerically.

c) Do the simulation and check that your prediction in (b) is correct.

6.5 We consider the Fibonacci equation with nonstandard initial values

$$x_{n+2} - x_{n+1} - x_n = 0, \quad x_0 = 0,\ x_1 = (1-\sqrt{5})/2. \tag{6.35}$$

a) Show that the general solution of the equation is

$$x_n = C\frac{1+\sqrt{5}}{2} + D\frac{1-\sqrt{5}}{2},$$

and that the initial values picks out the solution

$$x_n = \frac{1-\sqrt{5}}{2}.$$

b) What will happen if you simulate (6.35) on a computer?

c) Do the simulation and check that your predictions are correct.

6.6 We have the difference equation

$$x_{n+2} - \frac{2}{5}x_{n+1} + \frac{1}{45} = 0, \quad x_0 = 1,\ x_1 = 1/15. \tag{6.36}$$

a) Determine the general solution of (6.36) as well as the solution selected by the initial condition.

b) Why must you expect problems when you do a numerical simulation of the equation?

c) Determine approximately the value of n when the numerical solution has lost all significant digits.

d) Perform the numerical simulation and check that your predictions are correct.

6.7 In this exercise we consider the difference equation

$$x_{n+2} - \frac{5}{2}x_{n+1} + x_n = 0, \quad x_0 = 1,\ x_1 = 1/2.$$

a) Determine the general solution, and the solution corresponding to the initial conditions.

b) What kind of behaviour do you expect if you simulate the equation numerically?

c) Do the simulation and explain your results.