

## Stability

A formula (or more generally an algorithm) for solving a problem is called *unstable* if rounding errors during its computation cause a large error in the result. The above formula for  $r_1$  may be unstable when  $b \geq 0$  and  $b^2$  is much larger than  $|4ac|$ .

We can get a stable algorithm for  $r_1$  by noticing that if there is cancellation in  $-b + \sqrt{b^2 - 4ac}$  then there isn't in  $-b - \sqrt{b^2 - 4ac}$ . So first we calculate  $r_2$ . Then we notice that

$$ax^2 + bx + c = a[(x - r_1)(x - r_2)]$$

so  $r_1 r_2 = c/a$ , and so we can use the formula  $r_1 = c/ar_2$  to calculate  $r_1$  stably.

## Conditioning

When we want to solve the equation  $ax^2 + bx + c = 0$ , the numbers  $a, b$  and  $c$  usually have some error to begin with, because they likely come from another calculation or some measurement.

So when we use a stable algorithm to solve the equation, we're getting a relatively correct answer to a somewhat incorrect question.

We must consider how much effect error in the coefficients of  $ax^2 + bx + c$  has on the roots. For simplicity, let's look at the special case of calculating the positive solution to  $x^2 - c = 0$  for  $c > 0$ : this has the solution  $x = \sqrt{c}$ .

Suppose the true value of  $c$  is 0.25 and we use  $c' = 0.36$  instead. Then we're calculating 0.6 instead of 0.5. The relative error in the input is  $0.11/0.25 = 0.44$ , and the relative error in the result is  $0.1/0.5 = 0.2$ . Taking the square root has reduced the relative error by about a factor of 2.

The relative error in the result relative to the relative error in the input is called the condition number. For our problem it's:

$$\begin{aligned} \left| \frac{\frac{\sqrt{c'} - \sqrt{c}}{\sqrt{c}}}{\frac{c' - c}{c}} \right| &= \left| \sqrt{c} \frac{\sqrt{c'} - \sqrt{c}}{c' - c} \right| \\ &= \sqrt{c} \frac{1}{\sqrt{c'} + \sqrt{c}}. \end{aligned}$$

What's nice about this problem is that the condition number is at most 1. So the relative error in the result is no more than the relative error in the input.

In fact, if  $c'$  is close to  $c$  the condition number is close to 1/2: taking the square root reduces the relative error.

When calculating a function  $f$  using  $x'$  instead of  $x$ , the condition number is

$$\left| \frac{f(x') - f(x)}{f(x)} \right| / \left| \frac{x' - x}{x} \right|.$$

This depends on  $x'$  and  $x$ , but we can take the limit as  $x' \rightarrow x$  to get an approximation of the condition number for  $x'$  near  $x$ :

$$\begin{aligned} \lim_{x' \rightarrow x} \left| \frac{f(x') - f(x)}{f(x)} \right| / \left| \frac{x' - x}{x} \right| &= \left| \frac{x}{f(x)} \right| \lim_{x' \rightarrow x} \left| \frac{f(x') - f(x)}{x' - x} \right| \\ &= \left| \frac{x f'(x)}{f(x)} \right|, \end{aligned}$$

assuming  $f'(x)$  exists and  $f(x) \neq 0$ .

For  $f(x) = \sqrt{x}$ , this limit is

$$\left| \frac{x(1/2)x^{-1/2}}{x^{1/2}} \right| = 1/2.$$

Not all problems have such a nice effect on the error in their input. Consider calculating  $\tan x$  for some  $x$  near  $\pi/2$ :

$$\begin{aligned} \tan(1.57078) &\approx 6.12490 \times 10^4 \\ \tan(1.57079) &\approx 1.58058 \times 10^5 \end{aligned}$$

so a relative change in input of less than  $10^{-5}$  caused a relative change in output of more than 2. The condition number is more than  $2 \times 10^5$ !

## Conditioning versus stability

Stability is a property of algorithms. Condition is a property of problems.

A numerical algorithm can be viewed as a sequence of calculations solving certain subproblems. If one of the calculations is meant to solve a subproblem that has large condition number, then it will amplify errors in its input and likely (question: have we seen an example where this didn't happen?) lead to an unstable algorithm.

## Truncation

We've been talking about rounded results of a step as if the true value is calculated and then rounded. But can we get the true value, in a reasonable amount of time?

Consider adding 9876.5 and 7.6543 in the system  $\beta = 10$  and  $p = 5$ . It's best to start by rounding the number with smaller exponent according to the exponent of the larger: 7.6543 to 7.7. Then we do the addition  $9876.5 + 7.7 = 9884.2$ . Notice we never calculated the true value (think how wasteful it would be if we did so for  $1 \times 10^{100} + 1 \times 10^{-100}$ ).

For division, it's even more important not to generate the true value: it may have an infinite number of digits! But from elementary school we know a way (long division) to generate any number of initial digits without producing the rest.

What about  $\sqrt{x}$ ? For  $|x| < 1$ , one way of calculating it from the operations of addition, subtraction, multiplication and division is by the Taylor series:

$$\sqrt{x} = 1 + \frac{1}{2}x - \frac{1}{8}x^2 + \frac{1}{16}x^3 - \frac{5}{128}x^4 + \dots$$

We can't calculate the whole series, so we must truncate it to some finite number of terms. Once again, we're not calculating the true value. For efficiency we might not calculate enough terms to know even the correct rounded value. The error we introduce by not calculating the true value is called truncation error. Calculus gives us a way to estimate this error in the case of Taylor series.

Iterative algorithms (such as summing a series) can suffer significant truncation error. They can also suffer from another problem: the accumulation of a large number of relatively small errors. For example, repeatedly adding 0.1 to 100 in a system with  $\beta = 10$  and  $p = 3$  we soon have a large relative error:  $100 + 0.1 + 0.1 + \dots + 0.1$  is calculated as 100 no matter how many terms we add!