Condition and Stability

In Numerical Analysis there are always two fundamental questions we should consider when solving a problem: how sensitive is the solution to the specific form of the problem we are solving and how sensitive is the algorithm we use to solve the problem to the accuracy of the values we use?

Condition of a Problem

Every problem that we try to solve is based on an expression of some form or another. To have confidence in our solution we would first need to know that the expression is continuous in its inputs, so that we would not get completely different results from slight changes in the input. However, this is not enough. We also need to know that the expression is *well-conditioned*. If it is well-conditioned, then small changes in the input to the expression, will lead to small changes in the results. If small changes in the input lead to large changes in the output, then we call the problem *ill-conditioned*. The exact cutoff between well- and ill-conditioned depends on the context of the problem and the uses of the results.

Example: Suppose we want to evaluate the expression $y = x/(1-x)$. With $x = 0.93$ we get $y = 13.28...$, but with $x = 0.94$ we get $y = 15.66....$ So we would probably say that this expression is ill-conditioned when evaluated for x near 0.93. On the other hand, if we use $x = -0.93$ and $x = -0.94$, we get values of $-0.4818...$ and $-0.4845...$ and we would say this it is well-conditioned for x near -0.93.

For many types of problems we can compute a *condition number* that indicates the magnification of the changes. The condition number is defined by:

$$\text{Relative error in the output} \approx \text{Condition number} \times \text{Relative error in the input.}$$

For example, consider evaluating a function $f(x)$ at a point $x = x_0$. The input is x_0 and the output is $f(x_0)$. If we perturb the input to $x = x_0 + \epsilon$ then the output is $f(x_0 + \epsilon)$ and by applying the Mean Value Theorem we get

$$\frac{f(x_0+\epsilon) - f(x_0)}{f(x_0)} = \frac{\epsilon f'(\xi)}{f(x_0)} \approx \left[\frac{x_0 f'(x_0)}{f(x_0)}\right]\left(\frac{\epsilon}{x_0}\right),$$

where ξ is between x_0 and $x_0+\epsilon$. So the condition number of f at x_0, is given approximately by

$$C_f(x_0) = \left|\frac{x_0 f'(x_0)}{f(x_0)}\right|.$$

Applying this to $f(x) = x/(1-x)$ we get $C_f(x) = \frac{1}{|1-x|}$. Then $C_f(0.93) = 14.28...$ and $C_f(-0.93) = 0.5181....$ This is consistent with what we saw in the example above.

With the condition number as defined above, we still have no sharp cutoff between well- and ill-conditioned. We do know that if the condition number is less than 1 then it is well-conditioned and if the condition number is arbitrarily large then it is ill-conditioned. We usually study condition in limiting cases where these extremes are observed. For example, $f(x) = x/(1-x)$ is clearly ill-conditioned for x near 1 and well-conditioned for $x < 0$ and $x > 2$.

Stability of an Algorithm

When we study an algorithm our interest is the same as for an expression: we want small changes in the input to only produce small changes in the output. An algorithm or numerical process is called *stable* if this is true and it is called *unstable* if large changes in the output are produced. Analyzing an algorithm for stability is more complicated than determining the condition of an expression, even if the algorithm simply evaluates the expression. This is because an algorithm consists of many basic calculations and each one must be analyzed and, due to roundoff error, we must consider the possibility of small errors being introduced in every computed value. For example evaluating $y = x/(1-x)$ would be broken into 2 steps: $t = 1 - x$ and $y = x/t$. Also, we would consider both x and t to have small errors.

An algorithm is *stable* if every step is well-conditioned. It is *unstable* if any step is ill-conditioned.

Example: Consider evaluating $f(x) = \sqrt{1+x} - 1$ for x near 0. $C_f(x) = \frac{\sqrt{1+x}+1}{2\sqrt{1+x}}$ so $C_f(0) = 1$ so it is not ill-conditioned. To compute this we would have the 3 steps: (1) $t_1 = 1 + x$, (2) $t_2 = \sqrt{t_1}$ and (3) $f(x) = t_2 - 1$. Steps (1) and (2) are well-conditioned (condition numbers 0 and $1/2$), while Step (3) is ill-conditioned. Thus we would say that this algorithm is unstable.

This last example is interesting because the problem is well-conditioned but the obvious algorithm used to evaluate it is unstable. What this means is that there is a different algorithm for evaluating the original expression which would be stable. In this case we can re-formulate our function as $f(x) = \frac{x}{\sqrt{1+x}+1}$ by multiplying the original function by $\frac{\sqrt{1+x}+1}{\sqrt{1+x}+1}$. This new version is equivalent to the original function but by evaluating it in the obvious way, we have a stable algorithm.

Summary

- Well-/Ill-Conditioned refers to the problem; Stable/Unstable refers to an algorithm or numerical process.
- If the problem is well-conditioned then there is a stable way to solve it.
- If the problem is ill-conditioned then there is no reliable way to solve it in a stable way.
- Mixing roundoff-error with an unstable process is a recipe for disaster.
- With exact arithmetic (no roundoff-error), stability is not a concern.