STABILITY OF AN ALGORITHM, FROM THEORY TO PRACTICE

MAURIZIO PAOLINI

> In teoria la pratica corrisponde alla teoria, in pratica no.
>
> Alfio Quarteroni

1. Preliminaries

The model problem that we choose in order to show concepts like "stability" and "well-conditioning" is the computation of $\log(1+x)$. We studied in class its condition number, and concluded that the problem is ill-conditioned **only** for $x \approx -1$. In particular it is well-conditioned for small values of x, i.e. $|x| << 1$, the case in which we are particularly interested.

When studying the condition number we assume that the elementary functions involved are computed exactly. This is not the case when we practically perform the computation with some device, either when manually performing the operations one by one with a pocket calculator or when the computation is part of a computer program.

Indeed, when we speak of "algorithm" we also conventionally take into account the errors generated by each elementary operation as a result of the roundoff to a machine number. As we saw, all generated errors are bounded by ϵ_M, a small constant that depends on the floating point system that we are using. We might have control on the floating point system when using a high level language on a computer (chosing between single or double precision, for example), but often the floating point system is given for a given device. This is the case of the pocket calculators.

An algorithm can be described as, for example,

$$\texttt{flt}(\log(1+x)) \tag{1}$$

by which we mean that in the expression given as argument to `flt` all elementary operations must be substituted by the corresponding approximated operation performed by the computer, moreover, the quantity x is itself approximated (by rounding) into a machine number. It is important to note that we are not allowed to apply the usual arithmetic properties of the elementary functions inside the expression, since the approximated counterpart of the elementary operations do not usually satisfy these properties. For example, the algorithm (1) above is **different** from the algorithm

$$\texttt{flt}\left(\frac{x\log(1+x)}{(1+x)-1}\right) \tag{2}$$

although the two expressions are mathematically equivalent (at least for $x \neq 0$).

As we saw, the first algorithm (1) is **unstable** for $|x| << 1$. It is ill-conditioned because the logarithm greatly amplifies the (small) error generated by the sum; on the contrary we already said that the problem is well-conditioned: $K_{\text{ALG}} >> K_f$.

The first algorithm is **stable** for all other values of x: $x >> 0$ and $x \approx -1$. What is the conditioning of the problem and of the algorithm in these two regimes?

Strangely (this was the first assignment of the homework) the second algorithm (2) is stable also in the critical regime $|x| << 1$.

Ok, This is the theory. But what happens in practice? Did we only make a weird theoretical analysis, or is this what **really** happens during real computations?

Remark 1.1. The algorithm (2) does not work when $\texttt{flt}(1+x) = 1$, which is possible even for nonzero (but very small) values of x. In this event the algorithm is required to simply return $\texttt{flt}(x)$ (can you justify this?) We shall however deal with cases when this does not happen.

2. Choosing the device

I grabbed an old pocket calculator and used it to actually perform the computations above. For the records: it is a Texas Instruments "TI-30Xa" model. It uses scientific notation (this is common for almost all pocket calculators) and in particular it is able to compute natural logarithms (the button labelled $\boxed{\texttt{LN}}$). It also has a few memory registers; they are useful to store the x value (registry 1) and the result of $\texttt{flt}(1+x)$ (registry 2).

Of course I do not expect that you posses the same calculator, however I urge you to perform the calculations shown below on various devices. They could be:

- a pocket calculator (having at least the possibility to compute the log);
- a desktop pc or a laptop, using a computer program, or through some computational environment, such as `matlab`, or `octave`, or `scilab` or others (`R`?);
- a desktop pc or a laptop, through an emulated calculator (such as `kcalc` of KDE;
- a smartphone, using an `app`;
- a web emulator.

Each choice has it's own floating point system, in particular a base β and a precision p, however we can safely assume that β is either 10 or 2 (or perhaps a power of 2), and that p is at least 10 if $\beta = 10$.

3. Choosing x

We are interested in the $|x| << 1$ regime, so we shall select a specific small nonzero value of $x > 0$. It cannot be too small, otherwise $\texttt{flt}(1+x) = 1$ and we cannot use algorithm (2). Moreover we want the rounding error of the sum to be nonzero, otherwise there would be no error to be amplified by the log (the ill-conditioned operation). In other words we want x such that its floating point representation is not finite (i.e. it has an infinite number of nonzero digits). Given that $\beta = 10$ or $\beta = 2$ (or a power of 2), a fraction like $1/3$ has such an infinite floating point representation.

Furthermore we can assume that the calculator has at least 10 decimal digits in its mantissa (for pocket calculators); ambients like those mentioned above all use double precision, with a machine error much smaller than that of a pocket calculator.

The choice

$$x = \frac{1}{3} \cdot 10^{-8} \tag{3}$$

satisfies the above criteria, so we shall stick with that choice.

4. Exact value

We could be tempted to simply ask a computer to compute the value of $\log(1+x)$ with x given by (3). Just to give an example, here is the result obtained using `python`

```
$ python
Python 2.7.10 (default, Sep  8 2015, 17:21:32)
[...]
>>> from math import *
>>> x=1./3. * 1e-8
>>> x
3.3333333333333334e-09
>>> log(1+x)
3.333333381534409e-09
```

One is tempted to take for granted all the displayed digits (exept possibly the last one), however the result is wrong already in its ninth decimal place (digit '8'). Indeed, the log function is concave (negative second derivative), so that the value is always below the value of the tangent at 1. In our case this means $\log(1+x) < x = 3.33333333333... \cdot 10^{-9}$. This inaccuracy is not surprising: it is exactly the instability of the first algorithm that is causing such a large error.

Since x is quite small, we can use the Taylor expansion

$$\log(1+x) = x - \frac{1}{2}x^2 + \frac{1}{3}x^3 + \dots$$

to obtain the much more accure value $x \approx 3.333333327777778 \cdot 10^{-9}$:

```
$ python
Python 2.7.10 (default, Sep  8 2015, 17:21:32)
[...]
>>> x=1.0/3.0*1e-8
>>> x-1.0/2.0*x*x+1.0/3.0*x*x*x
3.333333327777778e-09
```

the same result would have been obtained also with one less term in the Taylor expansion.

We can test with python the correctness of the second algorithm (2)

```
$ python
Python 2.7.10 (default, Sep  8 2015, 17:21:32)
[...]
>>> from math import *
>>> x = 1./3*1e-8
>>> x*log(1+x)/((1+x)-1)
3.3333333277777775e-09
```

and indeed we see an error only in the last place (the 16-th), which is compatible with the underlying double-precision floating point system and the stability of the second algorithm (2).

5. Using the TI-30Xa...

The sequence of keystrokes [3] [1/x] [×] [1] [EE] [8] [+/-] [=] produces for x a displayed value of $3.333333333 \cdot 10^{-9}$ which we store in the first memory register. It is quite common that the internal precision of a pocket computer is a

bit larger than what is apparent looking at the display, we can expect a few more digits of precision, usually two or three.

Without entering into the detail of keystroke sequences, here is the final result given by the two algorithms

$$\text{algorithm (1):}\quad 3.330366975 \cdot 10^{-9}$$

$$\text{algorithm (2):}\quad 3.333700676 \cdot 10^{-9}$$

The second algorithm is indeed slightly better than the first. However, stability of the algorithm should lead to a result with an error comparable with the machine error, which in our case means about 10 decimal places. On the contrary, we obtain only 4 correct digits!

6. Exercises

Problem for you: is there an explanation for this bad performance of the second algorithm?

Hint 1: Computing the expression $\texttt{flt}((1+x)-1)$ with the pocket calculator gives

$$\texttt{flt}((1+x)-1) \to 3.33 \cdot 10^{-9}$$

Can you deduce from this the values of β and p of the floating point representation used by the pocket calculator?

Hint 2: Compute the exact value (at least the first 12 significant digits) of $\log(1 + 3.33 \cdot 10^{-9})$.

Hint 3: Does the pocket calculator adhere to the `IEEE` standard about the computed value of the elementary operation log?

If possible, try to perform the same computations with a different device and compare the results with those obtained with the TI-30Xa.