Analysis of Numerical Methods I

MTH6610 – Section 001 – Fall 2017

Lecture notes: Algorithm stability

Friday September 27, 2019

These notes are not a substitute for class attendance. Their main purpose is to provide a lecture overview summarizing the topics covered.

Reading: Trefethen & Bau III, Lectures 14, 15

We have previously investigated (i) mathematical conditioning of a problem, and (ii) the rounding/truncation error introduced by finite-precision representation of numbers. We are now ready to discuss stabilty of numerical algorithms used to solve mathematical problems.

If $f : \mathbb{R}^n \to \mathbb{R}^m$ is the mathematical problem at hand, we hope that a numerical algorithm $\widetilde{f} : \mathbb{R}^n \to \mathbb{R}^m$ satisfies the property that the relative error it commits is not too large, viz., that

$$\frac{\left\| \widetilde{f}(x) - f(x) \right\|}{\|f(x)\|}$$

is small. A short digression: the definition of the norm $\|\cdot\|$ above can be arbitrary, so long as $m, n < \infty$. This fact hinges on a well-established result that any norm over a finite-dimensional space is equivalent to any other norm. I.e., if $\|\cdot\|$ and $\|\cdot\|_*$ are any two norms on $\mathbb{R}^n$, then

$$c\|x\| \leq \|x\|_* \leq C\|x\|, \qquad\qquad x \in \mathbb{R}^n,$$

where the constants c, C are strictly positive and independent of x, but may depend on n. This result essentially allows us to prove convergence in one norm, and use inequalities like the above to extend the result to any other norm. Therefore, we use $\|\cdot\|$ in the remaining to denote arbitrary norms on $\mathbb{R}^n$ and $\mathbb{R}^m$.

Floating-point representations of real numbers have relative errors of at most machine precision, ϵ_{mach}. Thus, we are hoping for algorithms that satisfy

$$\frac{\left\| \widetilde{f}(x) - f(x) \right\|}{\|f(x)\|} = \mathcal{O}(\epsilon_{\mathrm{mach}}).$$

This is a bit unrealistic (or even unreasonable) to request if the mathematical problem f is ill-conditioned. We therefore expect some dependence on the condition of f to surface.

One basic fact is that there are two approximations happening: first an exact value of x is truncated to $\widetilde{x}$, and then this $\widetilde{x}$ is fed into a numerically approximate algorithm for f. We express the cumulation of these two approximations in the algorithm $\widetilde{f}$. An application of the triangle inequality yields

$$\frac{\left\| \widetilde{f}(x) - f(x) \right\|}{\|f(x)\|} \leq \frac{\left\| \widetilde{f}(x) - f(\widetilde{x}) \right\|}{\|f(x)\|} + \frac{\|f(\widetilde{x}) - f(x)\|}{\|f(x)\|} \tag{1}$$

The second term can be handled with our notion of the conditioning of f. Treatment of the first term requires a definition.

Definition 1. "Forward" stability
An algorithm $\widetilde{f}$ is (forward) stable if, for all $x \in \mathbb{R}^n$, we have

$$\frac{\left\| \widetilde{f}(x) - f(\widetilde{x}) \right\|}{\| f(\widetilde{x}) \|} = \mathcal{O}(\epsilon_{\text{mach}}),$$

for some $\widetilde{x}$ satisfying $\|x - \widetilde{x}\| = \|x\| \mathcal{O}(\epsilon_{\text{mach}})$.

This definition expresses the idea that we recognize $\widetilde{f}$ first performs the approximation $x \mapsto \widetilde{x}$. Therefore, we should measure the error in $\widetilde{f}$ relative to $f(\widetilde{x})$. This idea spawns the quip, "a forward stable algorithm yields an approximate answer to a closely related question."

Much of numerical analysis is then devoted to showing that a numerical algorithm is forward stable, where one hopefully can prove that

$$\frac{\left\| \widetilde{f}(x) - f(\widetilde{x}) \right\|}{\| f(x) \|} \leq C \epsilon_{\text{mach}} \sup_y \kappa(f, y). \tag{2a}$$

The definition of the condition of f yields

$$\frac{\| f(x) - f(\widetilde{x}) \|}{\| f(x) \|} \leq \frac{\| x - \widetilde{x} \|}{\| x \|} \sup_y \kappa(f, y) \tag{2b}$$

Finally, floating-point representation satisfies

$$\frac{\| x - \widetilde{x} \|}{\| x \|} = \mathcal{O}(\epsilon_{\text{mach}}) \tag{2c}$$

Using algorithm stability (2a), problem conditioning (2b), and the finite-precision bound (2c), we see that our accuracy desideratum (1) is achieved with a bound scaling like $\epsilon_{\text{mach}} \sup_y \kappa(f, y)$. This process of estimation, using the definition of stability we have introduced, is called *forward error analysis.*

It turns out that proving forward stability is actually quite hard in pratice. There is a competing, often easier, strategy for error analysis.

Definition 2. Backward stability
An algorithm $\widetilde{f}$, which approximates f, is backward stable if, for each $x \in \mathbb{R}^n$, we have

$$\widetilde{f}(x) = f(\widetilde{x})$$

for some $\widetilde{x}$ satisfying (2c).

Hence, "a backward stable algorithm yields an exact answer to a closely related question." Proving accuracy with a backward stable algorithm is straightforward:

$$\frac{\left\| \widetilde{f}(x) - f(x) \right\|}{\| f(x) \|} = \frac{\| f(\widetilde{x}) - f(x) \|}{\| f(x) \|},$$

and this can be bounded by the conditioning of f (2b) and the precision bound (2c). The process of proving backward stability of $\widetilde{f}$ to achieve an accuracy bound is called *backward error analysis*.

In practice, it also turns out that proving that algorithms are backwards stable can be easier than proving that they are forward stable.