CS322 Lecture Notes: Condition numbers

Steve Marschner
Cornell University

11-16 April 2007

When we start to think about solving some numerical problem, we should be careful only to attempt to solve problems that actually have solutions. (It's surprisingly often that you can get some kind of useful answer to a problem that doesn't really have a solution, but this is a sure way to set yourself up for unexpected difficulties!) Being possible to solve numerically is a little different from being possible to solve analytically.

1 Ill-posed and ill-conditioned problems

In the 1920s Hadamard proposed a famous set of criteria for a problem to be *well posed*:

- The problem should have a solution.
- The solution should be unique.
- The solution should depend continuously on the data.

His context was mathematical physics, but the same idea applies in numerical computation. A pure mathematician may be happy with just #1 and #2, but if you have a solution that jumps around wildly as you change the data that goes into the problem, then you're sunk if the data is approximate (either because it was measured in the real world or because it was rounded off in a floating point number system).

A problem that meets Hadamard's conditions is known as *well posed*, and a problem that doesn't is *ill posed*.

Many interesting problems are ill posed when stated straightforwardly. For instance, suppose we drop a particle into a turbulent creek and look at where it crosses some line a little way downstream. Even if the experiment is perfectly repeatable and we can get the flow to be the same every time, the chaotic behavior will mean that tiny changes in initial position will cause completely different results, so that we can't hope to make meaningful individual predictions. (We might be able to make probabilistic predictions, though: maybe we can compute the distribution of ending locations for particles that begin in some finite interval of starting points.)

Often ill-posed problems are one sort or another of *inverse problem* in which we're basically trying to run some physical system backward. For example, given an out-of-focus photograph, you might like to determine the original, sharp scene that the camera was looking at. But changes in small details of the scene will be blurred away by the out-of-focus camera, so that information is lost and we can't hope to get it back. In this case the problem probably fails condition #2 because there are many scenes, differing only in tiny details, that would lead to the same out-of-focus image.

In practical situations we care not just whether a problem is well-posed, technically, but also *how* well-posed it is—or whether it is sufficiently well posed that we can solve it using a particular method. The term for this question is problem *conditioning*: a well-conditioned problem is one where the answer is clearly determined by the data, and an ill-conditioned problem is one where tiny errors can cause large changes in the answer, so that the answer can be quite uncertain even when the data is good.

2 Conditioning in linear systems

Enough about terminology and generalities. Let's look at a particular kind of problem. Suppose we're solving a linear system

$$Ax = b$$

for x with A and b given, and we need some particular accuracy in x. How precisely do we need to know b? Or, alternatively, given a particular degree of uncertainty in b, how much uncertainty is there in x? Furthermore, if A is also uncertain, how does that enter into the problem? What we really want to know is how uncertainty in the problem—errors in A and b—propagates into the solution x.

In this lecture we are interested in the worst-case error; in later lectures we'll ask some more specific questions about the statistical distribution of the errors.

[Before we go on: a refresher on vector and matrix norms. In this lecture we are in the business of finding bounds on errors, and inequalities involving norms are the available tools. In particular, there is the triangle inequality for vector norms ($\|x + y\| \leq \|x\| + \|y\|$) and the definition of a matrix norm ($\|A\| = \max_x \|Ax\|/\|x\|$, so $\|Ax\| \leq \|A\| \, \|x\|$).]

Suppose we replace the right hand side b with $b + \delta b$ and solve the system

$$A(x + \delta x) = b + \delta b$$

to find the resulting error δx. Since $Ax = b$, we can cancel Ax with b and are left with

$$A\delta x = \delta b.$$

That is, the errors in x and b are related in the same way their values are. For a given b we can bound the worst-case error in δx:

$$\|\delta x\| = \|A^{-1}\delta b\| \leq \|A^{-1}\| \, \|\delta b\|$$

Since a matrix norm measures maximum stretching, or maximum magnification of the length of vectors, under a transformation, the norm of A^{-1} plays an important role because it limits the magnification of errors in b. What property of A does this measure? It is the maximum value of

$$\begin{aligned}\|A^{-1}\| &= \max_b \frac{\|A^{-1}b\|}{\|b\|} = \max_x \frac{\|x\|}{\|Ax\|} \\ &= 1 \Big/ \min_x \frac{\|Ax\|}{\|x\|}\end{aligned}$$

so it is also measuring the maximum shrinkage in a vector when multiplied by A. Note that the SVD gives us ready access to this number: it's the last singular value of A. So our bound is that errors in b are magnified by (at most) a factor of $1/\sigma_n$ (if we measure error in the 2-norm, which we will).

There's a weakness in this way of measuring error: if you change the units of measurement it will change the error bound. If A was based on measuring x in meters, and we switch to measuring it in millimeters, $\|A^{-1}\|$ will go up by a factor of 1,000. It's true that the numbers in x are more uncertain when they are measured in millimeters, but that doesn't really have any bearing on the "real" accuracy of the answer.

For this reason we usually use *relative error* for these kinds of arguments. Rather than ask about the errors δx and δb we ask about the errors relative to the magnitudes of their values:

$$\frac{\delta x}{\|x\|} \quad , \quad \frac{\delta b}{\|b\|}$$

The magnification factor for relative error is related to the "raw" magnification factor $\|A^{-1}\|$:

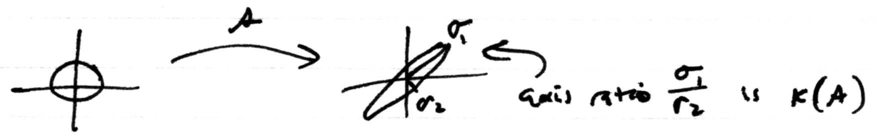
$$\begin{gathered}\|\delta x\| \leq \|A^{-1}\| \, \|\delta b\| \quad ; \quad \|b\| \leq \|A\|\|x\| \implies \frac{1}{\|x\|} \leq \frac{\|A\|}{\|b\|} \\ \text{so } \frac{\delta x}{\|x\|} \leq \|A\|\|A^{-1}\| \frac{\delta b}{\|b\|}\end{gathered}$$

So relative error is magnified by at most the product of the norms of A and its inverse. The norm of A^{-1} gives us an upper bound for the size of the error, and the norm of A gives us a lower bound for the size of x (we want a lower bound because the relative error in x gets larger when $\|x\|$ gets smaller.

Because relative error is the most generally used error measure, the magnification factor for relative error gets its own name, the *condition number*:

$$\kappa(A) = \|A\|\|A^{-1}\|$$

The SVD is a great tool for thinking about conditioning. Recall that $\|A\| = \sigma_1$, A's first (largest) singular value. This means that $\kappa(A) = \sigma_1/\sigma_n$. It's the multiplicative spread of the singular values. If we think of it geometrically, the condition number is the axis ratio that tells how flattened a sphere gets when we transform it by A:



Some quick facts about $\kappa(A)$:

$$\kappa(\alpha A) = \kappa(A)$$
$$\kappa(A^{-1}) = \kappa(A)$$
$$\kappa(A^T) = \kappa(A)$$

3 Propagation of errors from A to x

[To be added. Punch line: the condition number also bounds the magnitude of error in x relative to the magnitude of an error in A.]

Sources

- Our textbook: Moler, *Numerical Computing in Matlab.* Section 2.9.
- Golub and Van Loan, *Matrix Computations*, Third edition. Johns Hopkins Univ. Press, 1996. Section 2.7.
- Press, Flannery, Teukolsky, and Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Second Edition. Cambridge University Press, 1992. Chapter 2.
- Kress, *Numerical Analysis.* Springer-Verlag, 1998. Chapter 5.