Notes for 2015-02-06

Stability of LU and iterative refinement

Gaussian elimination with partial pivoting is almost always backward stable in practice, **but** there are examples where it fails to be backward stable. Remember that backward stability in this case means that if $\hat{L}$ and $\hat{U}$ are the computed factors, then

$$P(A+E) - \hat{L}\hat{U}, \quad \|E\| \leq C\epsilon_{\text{mach}}\|A\|$$

where C is some modest value that depends polynomially on the size of A.

It is possible to diagnose failure of backward stability by looking at the quantities appearing during the LU factorization. But a more useful trick is to look at the residual error[1]

$$r = b - A\hat{x}.$$

where $\hat{x}$ is an approximate solution computed from the LU factors that are actually stored in the machine. Note that

$$r = A(x - \hat{x})$$

and by combining the inequalities

$$\begin{aligned} \|\hat{x} - x\| &\leq \|A^{-1}\|\|r\| \\ \|b\| &\geq \|A\|\|x\| \end{aligned}$$

we have

$$\frac{\|hatx - x\|}{\|x\|} \leq \kappa(A)\frac{\|r\|}{\|b\|}.$$

Of course, if we compute the residual in ordinary floating point, we might be concerned that the computed residual mostly consists of rounding error. On the other hand, unlike Gaussian elimination, we have a backward error analysis for matrix multiplication that does not suffer from the uncomfortable

[1] Or should it be $A\hat{x} - b$? Doesn't really matter much, so long as I'm internally consistent. In any event, we're often concerned with the magnitude of the residual, and not the direction.

caveat "it is backward stable except rarely, when it it isn't." Also, the cost of computing the residual is $O(n^2)$, unlike the $O(n^3)$ cost to compute an LU factorization; hence, we might be willing to pay a little to compute the residual with extra precision.

What do we do if we have a factorization with a not-tiny backward error? After checking the residual to see that the error is unacceptably large, we might want a way of fixing the problem. One method for doing this is *iterative refinement*, which relies on the idea that a mediocre factorization may still provide a lot of value. The key to iterative refinement is the observation that if $\hat{x}$ is an approximate solution, then

$$A(x - \hat{x}) = r,$$

or $x = \hat{x} + A^{-1}r$. If we replace A^{-1} an approximation $\hat{A}^{-1}$ that comes from solving the system with an approximate factorization, we have the fixed point iteration

$$x^{(k+1)}) = x^{(k)} + \hat{A}^{-1}(b - Ax^{(k)}).$$

We have already looked at the analysis of fixed point iterations in 1D; here, the analysis is not much different. Subtract the fixed point equation

$$x = x + \hat{A}^{-1}(b - Ax),$$

and we find

$$e^{(k+1)} = (I - \hat{A}^{-1}A)e^{(k)}.$$

Taking norms, we find that the rate of convergence of the iteration depends on $\|I - \hat{A}^{-1}A\|$. In most cases, this is small enough for an approximate factorization that iterative refinement restores backward stability of the result in one step. Of course, just because the result is backward stable doesn't mean that the forward error will be small! Hence, for ill-conditioned problems it is sometimes a good idea to use iterative refinement in which the residuals are computed with extra precision.