Week 5: Monday, Sep 21

Block Gaussian elimination revisited

Though some of you might make use of cache blocking ideas in your own work, most of you will never try to write a cache-efficient Gaussian elimination routine of your own. The routines in LAPACK and MATLAB(really the same routines) are plenty efficient, so you would most likely turn to them. Still, it is worth knowing how to think about block Gaussian elimination, because sometimes the ideas can be specialized to build fast solvers for linear systems when there are fast solvers for sub-matrices

For example, consider the *bordered* matrix

$$A = \begin{bmatrix} B & W \\ V^T & C \end{bmatrix},$$

where B is an n-by-n matrix for which we have a fast solver and C is a p-by-p matrix, $p \ll n$. We can factor A into a product of *block* lower and upper triangular factors with a simple form:

$$\begin{bmatrix} B & W \\ V^T & C \end{bmatrix} = \begin{bmatrix} B & 0 \\ V^T & L_{22} \end{bmatrix} \begin{bmatrix} I & B^{-1}W \\ 0 & U_{22} \end{bmatrix}$$

where $L_{22}U_{22} = C - V^T A^{-1} W$ is an ordinary (small) factorization of the trailing Schur complement. To solve the linear system

$$\begin{bmatrix} B & W \\ V^T & C \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix},$$

we would then run block forward and backward substitution:

$$\begin{aligned} y_1 &= b_1 \\ y_2 &= L_{22}^{-1}\left(b_2 - V^T y_1\right) \end{aligned}$$

$$\begin{aligned} x_2 &= U_{22}^{-1} y_2 \\ x_1 &= y_1 - B^{-1}(W x_2) \end{aligned}$$

Perturbation theory

Recall the general strategy for error analysis that we described earlier: we want to derive forward error bounds by combining a sensitivity estimate (in terms of a *condition number*) with a *backward* error analysis that explains the computed result as the exact answer to a slightly erroneous problem. We will follow that strategy here, so it will be useful to work through the sensitivity analysis of solving linear systems.

Suppose that $Ax = b$ and that $\hat{A}\hat{x} = \hat{b}$, where $\hat{A} = A + \delta A$, $\hat{b} = b + \delta b$, and $\hat{x} = x + \delta x$. Then

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

Assuming the delta terms are small, we have the linear approximation

$$\delta A\, x + A\, \delta x \approx \delta b.$$

We can use this to get δx alone:

$$\delta x \approx A^{-1}(\delta b - \delta A\, x);$$

and taking norms gives us

$$\|\delta x\| \lesssim \|A^{-1}\|(\|\delta b\| + \|\delta A\|\|x\|).$$

Now, divide through by $\|x\|$ to get the relative error in x:

$$\frac{\|\delta x\|}{\|x\|} \lesssim \|A\|\|A^{-1}\| \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|A\|\|x\|} \right).$$

Recall that $\|b\| \leq \|A\|\|x\|$ to arrive at

$$\frac{\|\delta x\|}{\|x\|} \lesssim \kappa(A) \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|} \right),$$

where $\kappa(A) = \|A\|\|A^{-1}\|$. That is, the relative error in x is (to first order) bounded by the condition number times the relative errors in A and b.

Residual good!

The analysis in the previous section is pessimistic in that it gives us the worst-case error in δx for *any* δA and δb. But what if we are given data that behaves better than the worst case?

If we know A and b, a reasonable way to evaluate an approximate solution $\hat{x}$ is through the residual $r = b - A\hat{x}$. The approximate solution satisfies

$$A\hat{x} = b + r,$$

so if we subtract of $Ax = b$, we have

$$\hat{x} - x = A^{-1}r.$$

We can use this to get the error estimate

$$\|\hat{x} - x\| = \|A^{-1}\|\|r\|;$$

but for a given $\hat{x}$, we also actually have a prayer of *evaluating* $\delta x = A^{-1}r$ with at least some accuracy. It's worth pausing to think how novel this situation is. Generally, we can only *bound* error terms. If I tell you "my answer is off by just about 2.5," you'll look at me much more sceptically than if I tell you "my answer is off by no more than 2.5," and reasonably so. After all, if I knew that my answer was off by nearly 2.5, why wouldn't I add 2.5 to my original answer in order to get something closer to truth? This is exactly the idea behind *iterative refinement*:

1. Get an approximate solution $A\hat{x}_1 \approx b$.
2. Compute the residual $r = b - A\hat{x}_1$ (to good accuracy).
3. Approximately solve $A\,\delta x_1 \approx r$.
4. Get a new approximate solution $\hat{x}_2 = \hat{x}_1 + \delta x_1$; repeat as needed.