Week 5: Wednesday, Sep 23

Sherman-Morrison-Woodbury

The Sherman-Morrison formula describes the solution of $A+uv^T$ when there is already a factorization for A. An easy way to derive the formula is through block Gaussian elimination. In order to compute the product $(A+uv^T)x$, we would usually first compute $\xi = v^T x$ and then compute $(A+uv^T)x = Ax+u\xi$. So we can write $(A+uv^T)x = b$ in terms of an extended linear system

$$\begin{bmatrix} A & u \\ v^T & -1 \end{bmatrix} \begin{bmatrix} x \\ \xi \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}.$$

We can factor the matrix in this extended system as

$$\begin{bmatrix} A & u \\ v^T & -1 \end{bmatrix} = \begin{bmatrix} I & 0 \\ v^T A^{-1} & 1 \end{bmatrix} \begin{bmatrix} A & u \\ 0 & -1 - v^T A^{-1} u \end{bmatrix},$$

apply forward substitution with the block lower triangular factor,

$$\begin{aligned} y &= b, \\ \eta &= -v^T A^{-1} y, \end{aligned}$$

and apply backward substitution with the block upper triangular factor,

$$\begin{aligned} \xi &= (-1 - v^T A^{-1} u)^{-1} \eta \\ x &= A^{-1}(y - u\xi). \end{aligned}$$

If we put all the algebra together, we find

$$x = \left[A^{-1} - \frac{A^{-1} u v^T A^{-1}}{1 + v^T A^{-1} u} \right] b,$$

or

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1} u v^T A^{-1}}{1 + v^T A^{-1} u}.$$

This last formula is usually called the Sherman-Morrison formula. The Sherman-Morrison-Woodbury formula is the generalization to a rank k modification to A:

$$(A + UV^T)^{-1} = A^{-1} - A^{-1} U (I + V^T A^{-1} U)^{-1} V^T A^{-1}.$$

Backward error in Gaussian elimination

In lecture, I cut straight to the point: solving $Ax = b$ in finite precision using Gaussian elimination followed by forward and backward substitution yields a computed solution $\hat{x}$ *exactly* satisfies

$$(A + \delta A)\hat{x} = b, \tag{1}$$

where $|\delta A| \lesssim 3n\epsilon_{\text{mach}}|\hat{L}||\hat{U}|$, assuming $\hat{L}$ and $\hat{U}$ are the computed L and U factors.

Though I didn't do this in class, here I will briefly sketch a part of the error analysis following Demmel's treatment (§2.4.2). Mostly, this is because I find the treatment in §3.3.1 of our book less clear than I would like – but also, the bound in Demmel's book is marginally tighter. Here is the idea. Suppose $\hat{L}$ and $\hat{U}$ are the computed L and U factors. We obtain $\hat{u}_{jk}$ by repeatedly subtracting $l_{ji}u_{ik}$ from the original a_{jk}, i.e.

$$\hat{u}_{jk} = \text{fl}\left(a_{jk} - \sum_{i=1}^{j-1} \hat{l}_{ji}\hat{u}_{ik}\right).$$

Regardless of the order of the sum, we get an error that looks like

$$\hat{u}_{jk} = a_{jk}(1 + \delta_0) - \sum_{i=1}^{j-1} \hat{l}_{ji}\hat{u}_{ik}(1 + \delta_i) + O(\epsilon_{\text{mach}}^2)$$

where $|\delta_i| \leq (j-1)\epsilon_{\text{mach}}$. Turning this around gives

$$\begin{aligned} a_{jk} &= \frac{1}{1+\delta_0}\left(\hat{l}_{jj}\hat{u}_{jk} + \sum_{i=1}^{j-1} \hat{l}_{ji}\hat{u}_{ik}(1+\delta_i)\right) + O(\epsilon_{\text{mach}}^2) \\ &= \hat{l}_{jj}\hat{u}_{jk}(1-\delta_0) + \sum_{i=1}^{j-1} \hat{l}_{ji}\hat{u}_{ik}(1+\delta_i-\delta_0) + O(\epsilon_{\text{mach}}^2) \\ &= \left(\hat{L}\hat{U}\right)_{jk} + E_{jk}, \end{aligned}$$

where

$$E_{jk} = -\hat{l}_{jj}\hat{u}_{jk}\delta_0 + \sum_{i=1}^{j-1} \hat{l}_{ji}\hat{u}_{ik}(\delta_i - \delta_0) + O(\epsilon_{\text{mach}}^2)$$

is bounded in magnitude by $(j-1)\epsilon_{\text{mach}}(|L||U|)_{jk} + O(\epsilon_{\text{mach}}^2)$[1]. A similar argument for the components of $\hat{L}$ yields

$$A = \hat{L}\hat{U} + E, \text{ where } |E| \leq n\epsilon_{\text{mach}}|\hat{L}||\hat{U}| + O(\epsilon_{\text{mach}}^2).$$

In addition to the backward error due to the computation of the LU factors, there is also backward error in the forward and backward substitution phases, which gives the overall bound (1).

Pivoting

The backward error analysis in the previous section is not completely satisfactory, since $|L||U|$ may be much larger than $|A|$, yielding a large backward error overall. For example, consider the matrix

$$A = \begin{bmatrix} \delta & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \delta^{-1} & 1 \end{bmatrix} \begin{bmatrix} \delta & 1 \\ 0 & 1-\delta^{-1} \end{bmatrix}.$$

If $0 < \delta \ll 1$ then $\|L\|_\infty \|U\|_\infty \approx \delta^{-2}$, even though $\|A\|_\infty \approx 2$. The problem is that we ended up subtracting a huge multiple of the first row from the second row because δ is close to zero — that is, the leading principle minor is *nearly* singular. If δ were exactly zero, then the factorization would fall apart even in exact arithmetic. The solution to the woes of singular and near singular minors is pivoting; instead of solving a system with A, we re-order the equations to get

$$\hat{A} = \begin{bmatrix} 1 & 1 \\ \delta & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \delta & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1-\delta \end{bmatrix}.$$

Now the triangular factors for the re-ordered system matrix $\hat{A}$ have very modest norms, and so we are happy. If we think of the re-ordering as the effect of a permutation matrix P, we can write

$$A = \begin{bmatrix} \delta & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \delta & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1-\delta \end{bmatrix} = P^T LU.$$

[1] It's obvious that $E_j k$ is bounded in magnitude by $2(j-1)\epsilon_{\text{mach}}(|L||U|)_{jk} + O(\epsilon_{\text{mach}}^2)$. We cut a factor of two if we go down to the level of looking at the individual rounding errors during the dot product, because some of those errors cancel.

Note that this is equivalent to writing $PA = LU$ where P is another permutation (which undoes the action of P^T).

If we wish to control the multipliers, it's natural to choose the permutation P so that each of the multipliers is at most one. This standard choice leads to the following algorithm:

```
for j = 1:n-1

  % Find ipiv >= j to maximize |A(i,j)|
  [absAij, ipiv] = max(abs(A(j:n,j)));
  ipiv = ipiv + j-1;

  % Swap row ipiv and row j
  Aj = A(j,j:n);
  A(j,j:n) = A(ipiv,j:n);
  A(ipiv,j:n) = Aj;

  % Record the pivot row
  piv(j) = ipiv;

  % Update trailing submatrix
  A(j+1:n,j+1:n) = A(j+1:n,j+1:n) - A(j+1:n,j)*A(j,j+1:n);

end
```

By design, this algorithm produces an L factor in which all the elements are bounded by one. But what about the U factor? There exist pathological matrices for which the elements of U grow exponentially with n. But these examples are extremely uncommon in practice, and so, in general, Gaussian elimination with partial pivoting does indeed have a small backward error. Of course, the pivot growth is something that we can monitor, so in the unlikely event that it *does* look like things are blowing up, we can tell there is a problem and try something different.

When problems do occur, it is more frequently the result of ill-conditioning in the problem than of pivot growth during the factorization.

Iterative refinement revisited

Recall from last lecture that if we have a solver for $\hat{A} = A + E$ with E small, then we can use *iterative refinement* to "clean up" the solution. The matrix $\hat{A}$ could come from finite precision Gaussian elimination of A, for example, or from some factorization of a nearby "easier" matrix. To get the refinement iteration, we take the equation

$$Ax = \hat{A}x - Ex = b, \tag{2}$$

and think of x as the fixed point for an iteration

$$\hat{A}x_{k+1} - Ex_k = b. \tag{3}$$

Note that this is the same as

$$\hat{A}x_{k+1} - (\hat{A} - A)x_k = b,$$

or

$$x_{k+1} = x_k + \hat{A}^{-1}(b - Ax_k).$$

Note that this latter form is the same as inexact Newton iteration on the equation $Ax_k - b = 0$ with the approximate Jacobian $\hat{A}$.

If we subtract (2) from (3), we see

$$\hat{A}(x_{k+1} - x) - E(x_k - x) = 0,$$

or

$$x_{k+1} - x = \hat{A}^{-1}E(x_k - x).$$

Taking norms, we have

$$\|x_{k+1} - x\| \leq \|\hat{A}^{-1}E\|\|x_k - x\|.$$

Thus, if $\|\hat{A}^{-1}E\| < 1$, we are guaranteed that $x_k \to x$ as $k \to \infty$. At least, this is what happens in exact arithmetic. In practice, the residual is usually computed with only finite precision, and so we might expect to stop making progress at some point. This is the topic of the first problem in HW 2.