2022-09-13

1 Diagonally dominant matrices

A matrix A is *strictly (column) diagonally dominant* if for each column j,

$$|a_{jj}| > \sum_{i \neq j} |a_{ij}|.$$

If we write $A = D + F$ where D is the diagonal and F the off-diagonal part, strict column diagonal dominance is equivalent to the statement that

$$\|FD_{-1}\|_1 < 1.$$

Note that we may factor A as

$$A = (I + FD^{-1})D$$

with D invertible because the diagonal elements are bounded below by zero and $I+FD^{-1}$ invertible by a Neumann series bound. Therefore A is invertible if it is strictly column diagonally dominant.

Strict diagonal dominance is a useful structural condition for several reasons: it ensures nonsingularity, it guarantees convergence of certain iterative methods (we will return to this later), and it guarantees that LU factorization can be done without pivoting. In fact, Gaussian elimination without partial pivoting is guaranteed not to even attempt pivoting! To see this, note that the statement is obvious for the first step: column diagonal dominance implies that a_{11} is the largest magnitude element in the first column. What does the Schur complement look like after one step of Gaussian elimination? By a short computation, it turns out that the Schur complement is again diagonally dominant (see GVL section 4.1.1).

Diagonally dominant matrices and symmetric positive definite matrices are the two major classes of matrices for which unpivoted Gaussian elimination is backward stable.

2 Symmetric matrices

2.1 Quadratic forms

A matrix A is symmetric if $A = A^T$. For each symmetric matrix A, there is an associated quadratic form $x^T Ax$. Even if you forgot them from our lightning review of linear algebra, you are likely familiar with quadratic forms from a multivariate calculus class, where they appear in the context of the second derivative test. One expands

$$F(x+u) = F(x) + F'(x)u + \frac{1}{2}u^T H(x) u + O(\|u\|^3),$$

and notes that at a stationary point where $F'(x) = 0$, the dominant term is the quadratic term. When H is *positive definite* or *negative definite*, x is a strong local minimum or maximum, respectively. When H is indefinite, with both negative and positive eigenvalues, x is a saddle point. When H is semi-definite, one has to take more terms in the Taylor series to determine whether the point is a local extremum.

If B is a nonsingular matrix, then we can write $x = By$ and $x^T Ax = y^T(B^T AB)y$. So an “uphill” direction for A corresponds to an “uphill” direction for $B^T AB$; and similarly with downhill directions. More generally, A and $B^T AB$ have the same *inertia*, where the inertia of a symmetric A is the triple

$$(\text{\# pos eigenvalues}, \text{\# zero eigenvalues}, \text{\# neg eigenvalues}).$$

Now suppose that $A = LU$, where L is unit lower triangular and U is upper triangular. If we let D be the diagonal part of U, we can write $A = LDM^T$, where L and M are both unit lower triangular matrices. Noting that $A^T = (LDM^T)^T = MDL^T = M(LD)^T$ and that the LU factorization of a matrix is unique, we find $M = L$ and $LD = DM^T = U$. Note that D has the same inertia as A.

The advantage of the LDL^T factorization over the LU factorization is that we need only compute and store one triangular factor, and so LDL^T factorization costs about half the flops and storage of LU factorization. We have the same stability issues for LDL^T factorization that we have for ordinary LU factorization, so in general we might compute

$$PAP^T = LDL^T,$$

where the details of various pivoting schemes are described in the book.

2.2 Positive definite matrices

A symmetric matrix is positive definite if $x^T Ax > 0$ for all nonzero x. If A is symmetric and positive definite, then $A = LDL^T$ where D has all positive elements (because A and D have the same inertia). Thus, we can write $A = (LD^{1/2})(LD^{1/2})^T = \hat{L}\hat{L}^T$. The matrix $\hat{L}$ is a Cholesky factor of A.

There are several useful properties of SPD matrices that we will use from time to time:

1. The inverse of an SPD matrix is SPD.

   Proof: If $x^T A x > 0$ for all $x \neq 0$, then we cannot have $Ax = 0$ for nonzero x. So A is necessarily nonsingular. Moreover,
   $$x^T A^{-1} x = (A^{-1}x)^T A (A^{-1}x)$$
   must be positive for nonzero x by positive-definiteness of A. Therefore, A^{-1} is SPD.

2. Any minor of an SPD matrix is SPD.

   Proof: Without loss of generality, let $M = A_{11}$. Then for any appropriately sized x,
   $$x^T M x = \begin{bmatrix} x \\ 0 \end{bmatrix}^T A \begin{bmatrix} x \\ 0 \end{bmatrix} > 0$$
   for $x \neq 0$. Therefore, M is positive definite.

3. Any Schur complement of an SPD matrix is SPD

   Proof: A Schur complement in A is the inverse of a minor of an inverse of A. By the two arguments above, this implies that any Schur complement of an SPD matrix is SPD.

4. If M is a minor of A, $\kappa_2(M) \leq \kappa_2(A)$.

   Proof: The largest and smallest singular values of an SPD matrix are the same as the largest and smallest eigenvalues; they can be written as
   $$\sigma_1(A) = \max_{\|x\|_2=1} x^T A x, \quad \sigma_{\min}(A) = \min_{\|x\|_2=1} x^T A x.$$
   Without loss of generality, let $M = A_{11}$. Then
   $$\sigma_1(M) = \max_{\|x\|_2=1} x^T M x = \max_{\|x\|_2=1} \begin{bmatrix} x \\ 0 \end{bmatrix}^T A \begin{bmatrix} x \\ 0 \end{bmatrix} \leq \max_{\|z\|_2=1} z^T A z = \sigma_1(A)$$
   and similarly $\sigma_{\min}(M) \geq \sigma_{\min}(A)$. The condition numbers are therefore
   $$\kappa_2(M) = \frac{\sigma_1(M)}{\sigma_{\min}(M)} \leq \frac{\sigma_1(A)}{\sigma_{\min}(A)} = \kappa_2(A).$$

5. If S is a Schur complement in A, $\kappa_2(S) \leq \kappa_2(A)$.

   Proof: This is left as an exercise.

3 Cholesky

The algorithm to compute the Cholesky factor of an SPD matrix is close to the Gaussian elimination algorithm. In the first step, we would write

$$\begin{bmatrix} a_{11} & a_{21}^T \\ a_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 \\ l_{21} & L_{22} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21}^T \\ 0 & L_{22}^T \end{bmatrix},$$

or

$$\begin{aligned} a_{11} &= l_{11}^2 \\ a_{21} &= l_{21} l_{11} \\ A_{22} &= L_{22} L_{22}^T + l_{21} l_{21}^T. \end{aligned}$$

The first two equations allow us to compute the first column of L; the last equation tells us that the rest of L is the Cholesky factor of a Schur complement, $L_{22} L_{22}^T = A_{22} - l_{21} l_{21}^T$. Continuing in this fashion, we have the algorithm

```
#
# Overwrite with Cholesky factorization
#
function mychol(A)
    A = copy(A)
    n = size(A)[1]
    for j = 1:n
        if A[j,j] < 0.0
            error("Indefinite matrix")
        end
        A[j,j] = sqrt(A[j,j])
        A[j+1:end,j] /= A[j,j]
        A[j+1:end,j+1:end] -= A[j+1:end,j]*A[j+1:end,j]'
    end
    LowerTriangular(A)
end
```

Like the nearly-identical Gaussian elimination algorithm, we can rewrite the Cholesky algorithm in block form for better cache use. Unlike Gaussian elimination, we do just fine using Cholesky *without* pivoting[1].

4 Iterative refinement

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, possibly with a less stringent pivoting strategy than partial pivoting. Or it might come from some factorization of a nearby "easier" matrix. To get the refinement iteration, we take the equation

$$Ax = \hat{A}x - Ex = b, \tag{1}$$

and think of x as the fixed point for an iteration

$$\hat{A}x_{k+1} - Ex_k = b. \tag{2}$$

Note that this is the same as

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

or

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

If we subtract (1) from (2), 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 $k \to \infty$. In fact, this holds even if the backward error varies from step to step, as long as it satisfies some uniform bound that is less than one. At least, this is what happens in exact arithmetic.

[1] Pivoting can still be useful for near-singular matrices, but unpivoted Cholesky is backward stable

In practice, the residual is usually computed with only finite precision, and so we would stop making progress at some point — usually at the point where we have a truly backward stable solution. In general, iterative refinement is mainly used when either the residual can be computed with extra precision or when the original solver suffers from relatively large backward error.

In floating point arithmetic, we actually compute something like

$$x_{k+1} = x_k + \hat{A}_k^{-1}(b - Ax_k + \delta_k) + \mu_k,$$

where $\hat{A}_k = A + E_k$ accounts for the backward error E_k in the approximate solve, δ_k is an error associated with computing the residual, and μ_k is an error associated with the update. This gives us the error recurrence

$$e_{k+1} = \hat{A}_k^{-1} E_k e_k + \hat{A}^{-1}\delta_k + \mu_k$$

If $\|\delta_k\| < \alpha$, $\|\mu_k\| < \beta$, and $\|A^{-1}E_k\| < \gamma < 1$ for all k, then we can show that

$$\|x_k - x\| \leq \gamma^k \|x_0 - x\| + \frac{\alpha\|A^{-1}\| + \beta}{1-\gamma}.$$

If we evaluate the residual in the obvious way, we typically have

$$\begin{aligned} \alpha &\leq c_1 \epsilon_{\mathrm{mach}} \|A\| \|x\|, \\ \beta &\leq c_2 \epsilon_{\mathrm{mach}} \|x\|, \end{aligned}$$

for some modest c_1 and c_2; and for large enough k, we end up with

$$\frac{\|x_k - x\|}{\|x\|} \leq C_1 \epsilon_{\mathrm{mach}} \kappa(A) + C_2 \epsilon_{\mathrm{mach}}.$$

That is, iterative refinement leads to a relative error not too much greater than we would expect due to a small relative perturbation to A; and we can show that in this case the result is backward stable. And if we use *mixed* precision to evaluate the residual accurately enough relative to $\kappa(A)$ (i.e. $\alpha\kappa(A) \lesssim \beta$) we can actually achieve a small *forward* error.

5 Condition estimation

Suppose now that we want to compute $\kappa_1(A)$ (or $\kappa_\infty(A) = \kappa_1(A^T)$). The most obvious approach would be to compute A^{-1}, and then to evaluate

$\|A^{-1}\|_1$ and $\|A\|_1$. But the computation of A^{-1} involves solving n linear systems for a total cost of $O(n^3)$ — the same order of magnitude as the initial factorization. Error estimates that cost too much typically don't get used, so we want a different approach to estimating $\kappa_1(A)$, one that does not cost so much. The only piece that is expensive is the evaluation of $\|A^{-1}\|_1$, so we will focus on this.

Note that $\|A^{-1}x\|_1$ is a convex function of x, and that $\|x\|_1 \leq 1$ is a convex set. So finding

$$\|A^{-1}\|_1 = \max_{\|x\|_1 \leq 1} \|A^{-1}x\|_1$$

is a convex optimization problem. Also, note that $\|\cdot\|_1$ is differentiable almost everywhere: if all the components of y are nonzero, then

$$\xi^T y = \|y\|_1, \text{ for } \xi = \operatorname{sign}(y);$$

and if δy is small enough so that all the components of $y + \delta y$ have the same sign as the corresponding components of y, then

$$\xi^T (y + \delta y) = \|y + \delta y\|_1$$

More generally, we have

$$\xi^T u \leq \|\xi\|_\infty \|u\|_1 = \|u\|_1,$$

i.e. even when δy is big enough so that the linear approximation to $\|y+\delta y\|_1$ no longer holds, we at least have a lower bound.

Since $y = A^{-1}x$, we actually have that

$$|\xi^T A^{-1}(x + \delta x)| \leq \|A^{-1}(x + \delta x)\|,$$

with equality when δx is sufficiently small (assuming y has no zero components). This suggests that we move from an initial guess x to a new guess x_{new} by maximizing

$$|\xi^T A^{-1} x_{\text{new}}|$$

over $\|x_{\text{new}}\| \leq 1$. This actually yields $x_{\text{new}} = e_j$, where j is chosen so that the jth component of $z^T = \xi^T A^{-1}$ has the greatest magnitude.

Putting everything together, we have the following algorithm

```
function hager(n, solveA, solveAT)
    x = ones(n)/n
    invA_normest = 0.0
    while true

        y = solveA(x)  # Evaluate y = A^-1 x
        xi = sign.(y)  # and z = A^-T sign(y), the
        z = solveAT(xi) # subgradient of x -> |A^-1 x|_1

        # Find the largest magnitude component of z
        znorm, j = findmax(abs.(z))

        # Check for convergence
        if znorm <= dot(z,x)
            return norm(y,1)
        end

        # Update x to e_j and repeat
        x[:] .= 0.0
        x[j] = 1.0

    end
    invA_normest
end
```

This method is not infallible, but it usually gives estimates that are the right order of magnitude. There are various alternatives, refinements, and extensions to Hager's method, but they generally have the same flavor of probing A^{-1} through repeated solves with A and A^T.

6 Scaling

Suppose we wish to solve $Ax = b$ where A is ill-conditioned. Sometimes, the ill-conditioning is artificial because we made a poor choice of units, and it appears to be better conditioned if we write

$$D_1 A D_2 y = D_1 b,$$

where D_1 and D_2 are diagonal scaling matrices. If the original problem was poorly scaled, we will likely find $\kappa(D_1 A D_2) \ll \kappa(A)$, which may be great for Gaussian elimination. But by scaling the matrix, we are really changing the norms that we use to measure errors — and that may not be the right thing to do.

For physical problems, a good rule of thumb is to non-dimensionalize before computing. The non-dimensionalization will usually reveal a good scaling that (one hopes) simultaneously is appropriate for measuring errors and does not lead to artificially inflated condition numbers.