Mathematical Tripos Part IB: Lent 2020

Numerical Analysis – Lecture 14[1]

Sparse matrices It is often required to solve *very* large systems $A\boldsymbol{x} = \boldsymbol{b}$ ($n = 10^5$ is considered small in this context!) where nearly all the elements of A are zero. Such a matrix is called *sparse* and efficient solution of $A\boldsymbol{x} = \boldsymbol{b}$ should exploit sparsity. In particular, we wish the matrices L and U to inherit as much as possible of the sparsity of A and for the cost of computation to be determined by the number of nonzero entries, rather than by n. The following theorem shows that certain zeros of A are always inherited by an LU factorization.

Theorem Let $A = LU$ be an LU factorization (without pivoting) of a sparse matrix. Then all leading zeros in the rows of A to the left of the diagonal are inherited by L and all the leading zeros in the columns of A above the diagonal are inherited by U.

Proof We assume that $U_{k,k} \neq 0$ for all $k = 1, \ldots, n$ which is the same as saying that $(A_{k-1})_{k,k} \neq 0$ when running the LU factorization algorithm (without pivoting). If $A_{i,1} = 0$ this means that $L_{i,1}U_{1,1} = 0$ and so $L_{i,1} = 0$. If furthermore $A_{i,2} = 0$ we get $L_{i,1}U_{1,2} + L_{i,2}U_{2,2} = 0$ which implies $L_{i,2} = 0$ since $L_{i,1} = 0$. In general we get that if $A_{i,1} = \cdots = A_{i,j} = 0$ where $j < i$ then $L_{i,1} = \cdots = L_{i,j} = 0$. A similar reasoning applies for leading zeros in the columns of A above the diagonal. □

Banded matrices The matrix A is a *banded matrix* if there exists an integer $r < n$ such that $A_{i,j} = 0$ for $|i - j| > r$, $i, j = 1, 2, \ldots, n$. In other words, all the nonzero elements of A reside in a band of width $2r + 1$ along the main diagonal. In that case, according to the previous theorem, $A = LU$ implies that $L_{i,j} = U_{i,j} = 0$ $\forall\ |i - j| > r$ and sparsity structure is inherited by the factorization.
In general, the expense of calculating an LU factorization of an $n \times n$ *dense* matrix A is $\mathcal{O}(n^3)$ operations and the expense of solving $A\boldsymbol{x} = \boldsymbol{b}$, provided that the factorization is known, is $\mathcal{O}(n^2)$. However, in the case of a banded A, we need just $\mathcal{O}(r^2 n)$ operations to factorize and $\mathcal{O}(rn)$ operations to solve a linear system. If $r \ll n$ this represents a very substantial saving!

General sparse matrices feature a wide range of applications, e.g. the solution of partial differential equations, and there exists a wealth of methods for their solution. One approach is efficient factorization, that minimizes *fill-in* (a fill-in is an zero entry of the matrix A that gets *filled in* during the factorization, i.e., $A_{ij} = 0$ and yet $L_{ij} \neq 0$ (if $i > j$) or $U_{ij} \neq 0$ (if $j > i$)). Yet another is to use iterative methods (cf. Part II Numerical Analysis course). There also exists a substantial body of other, highly effective methods, e.g. Fast Fourier Transforms, preconditioned conjugate gradients and multigrid techniques (cf. Part II Numerical Analysis course), fast multipole techniques and much more.

Sparsity and graph theory An exceedingly powerful (and beautiful) methodology of ordering pivots to minimize fill-in of sparse matrices uses graph theory and, like many other cool applications of mathematics in numerical analysis, is alas not in the schedules `:-(`

5.2 QR factorization of matrices

Scalar products, norms and orthogonality We first recall a few definitions. $\mathbb{R}^n$ is the linear space of all real n-tuples.

- For all $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$ we define the *scalar product*

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \langle \boldsymbol{v}, \boldsymbol{u} \rangle = \sum_{j=1}^{n} u_j v_j = \boldsymbol{u}^\top \boldsymbol{v} = \boldsymbol{v}^\top \boldsymbol{u}\,.$$

- The vectors $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_m \in \mathbb{R}^n$ are *orthonormal* if

$$\langle \boldsymbol{q}_k, \boldsymbol{q}_\ell \rangle = \begin{cases} 1, & k = \ell, \\ 0, & k \neq \ell, \end{cases} \qquad k, \ell = 1, 2, \ldots, m.$$

[1] Corrections and suggestions to these notes should be emailed to `h.fawzi@damtp.cam.ac.uk`.

- An $n \times n$ real matrix Q is *orthogonal* if all its columns are orthonormal. Since $(Q^\top Q)_{k,\ell} = \langle \boldsymbol{q}_k, \boldsymbol{q}_\ell \rangle$, this implies that $Q^\top Q = I$ (I is the *unit matrix*). Hence $Q^{-1} = Q^\top$ and $QQ^\top = QQ^{-1} = I$. We conclude that the rows of an orthogonal matrix are also orthonormal, and that $Q^\top$ is an orthogonal matrix. Further, $1 = \det I = \det(QQ^\top) = \det Q \det Q^\top = (\det Q)^2$, and thus we deduce that $\det Q = \pm 1$, and that an orthogonal matrix is nonsingular.

The QR factorization The QR factorization of an $m \times n$ matrix A has the form $A = QR$, where Q is an $m \times m$ *orthogonal* matrix and R is an $m \times n$ *upper triangular* matrix (i.e., $R_{i,j} = 0$ for $i > j$). When $m \geq n$, a *reduced QR factorization* of A is a factorization $A = QR$ where Q is $m \times n$ with orthonormal columns, and R is $n \times n$ upper triangular.

Application in linear system solving Let $m = n$ and A be nonsingular. We can solve $A\boldsymbol{x} = \boldsymbol{b}$ by calculating the QR factorization of A and solving first $Q\boldsymbol{y} = \boldsymbol{b}$ (hence $\boldsymbol{y} = Q^\top \boldsymbol{b}$) and then $R\boldsymbol{x} = \boldsymbol{y}$ (a triangular system!).

Interpretation of the QR factorization Let $m \geq n$ and denote the columns of A and Q by $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n$ and $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n$ respectively. In a reduced QR factorization:

$$\begin{bmatrix} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_n \end{bmatrix} = \begin{bmatrix} \boldsymbol{q}_1 & \boldsymbol{q}_2 & \cdots & \boldsymbol{q}_n \end{bmatrix} \begin{bmatrix} R_{1,1} & R_{1,2} & \cdots & R_{1,n} \\ 0 & R_{2,2} & & \vdots \\ \vdots & \ddots & \ddots & \\ & & 0 & R_{n,n} \end{bmatrix},$$

we have $\boldsymbol{a}_k = \sum_{j=1}^{k} R_{j,k} \boldsymbol{q}_j$, $k = 1, 2, \ldots, n$. In other words, Q has the property that each kth column of A can be expressed as a linear combination of the first k columns of Q.

The Gram–Schmidt algorithm Assume that $m \geq n$ and that the columns of A are linearly independent. We will see how to construct a reduced QR factorization of A, i.e., $Q \in \mathbb{R}^{m \times n}$ having orthonormal columns, $R \in \mathbb{R}^{n \times n}$ upper-triangular and $A = QR$: in other words,

$$\sum_{k=1}^{\ell} R_{k,\ell} \boldsymbol{q}_k = \boldsymbol{a}_\ell, \quad \ell = 1, 2, \ldots, n, \qquad \text{where} \qquad A = \begin{bmatrix} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_n \end{bmatrix}. \tag{5.2}$$

Equation (5.2) for $\ell = 1$ tells us that we must have $\boldsymbol{q}_1 = \boldsymbol{a}_1 / \|\boldsymbol{a}_1\|$ and $R_{1,1} = \|\boldsymbol{a}_1\|$. Next we form the vector $\boldsymbol{b} = \boldsymbol{a}_2 - \langle \boldsymbol{q}_1, \boldsymbol{a}_2 \rangle \boldsymbol{q}_1$. It is orthogonal to $\boldsymbol{q}_1$, since $\langle \boldsymbol{q}_1, \boldsymbol{a}_2 - \langle \boldsymbol{q}_1, \boldsymbol{a}_2 \rangle \boldsymbol{q}_1 \rangle = \langle \boldsymbol{q}_1, \boldsymbol{a}_2 \rangle - \langle \boldsymbol{q}_1, \boldsymbol{a}_2 \rangle \langle \boldsymbol{q}_1, \boldsymbol{q}_1 \rangle = 0$. Since the columns of A are assumed linearly independent, $\boldsymbol{b} \neq \boldsymbol{0}$ and we set $\boldsymbol{q}_2 = \boldsymbol{b} / \|\boldsymbol{b}\|$, hence $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ are orthonormal. Moreover,

$$\langle \boldsymbol{q}_1, \boldsymbol{a}_2 \rangle \boldsymbol{q}_1 + \|\boldsymbol{b}\| \boldsymbol{q}_2 = \langle \boldsymbol{q}_1, \boldsymbol{a}_2 \rangle \boldsymbol{q}_1 + \boldsymbol{b} = \boldsymbol{a}_2,$$

hence, to obey (5.2) for $\ell = 2$, we let $R_{1,2} = \langle \boldsymbol{q}_1, \boldsymbol{a}_2 \rangle$, $R_{2,2} = \|\boldsymbol{b}\|$.

More generally we get the following classical Gram-Schmidt algorithm to compute a QR factorization: Set $\boldsymbol{q}_1 = \boldsymbol{a}_1 / \|\boldsymbol{a_1}\|$ and $R_{11} = \|\boldsymbol{a_1}\|$. For $j = 2, \ldots, n$: Set $R_{ij} = \langle \boldsymbol{q}_i, \boldsymbol{a}_j \rangle$ for $i \leq j - 1$, and $\boldsymbol{b}_j = \boldsymbol{a}_j - \sum_{i=1}^{j-1} R_{ij} \boldsymbol{q}_i$. Set $\boldsymbol{q}_j = \boldsymbol{b}_j / \|\boldsymbol{b}_j\|$ and $R_{jj} = \|\boldsymbol{b}_j\|$.

The total cost of the classical Gram–Schmidt algorithm is $\mathcal{O}\big(n^2 m\big)$, since at each iteration j a total of $\mathcal{O}(mj)$ operations are performed.

The disadvantage of the classical Gram–Schmidt is its *ill-conditioning*: using finite arithmetic, small imprecisions in the calculation of inner products spread rapidly, leading to effective loss of orthogonality. Errors accumulate fast and the computed off-diagonal elements of $Q^\top Q$ may become large.