Jim Lambers
CME 335
Spring Quarter 2010-11
Lecture 7 Notes

Jacobi Methods

One of the major drawbacks of the symmetric QR algorithm is that it is not parallelizable. Each orthogonal similarity transformation that is needed to reduce the original matrix A to diagonal form is dependent upon the previous one. In view of the evolution of parallel architectures, it is therefore worthwhile to consider whether there are alternative approaches to reducing an $n \times n$ symmetric matrix A to diagonal form that can exploit these architectures.

The Jacobi Idea

To that end, we consider a rotation matrix, denoted by $J(p, q, \theta)$, of the form

$$\begin{array}{c} \begin{bmatrix} 1 & \cdots & 0 & & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & & 0 & \cdots & 1 \end{bmatrix} \\ \begin{matrix} & & p & & q & & \end{matrix} \end{array} \begin{matrix} \\ \\ p \\ \\ q \\ \\ \\ \end{matrix}$$

where $c = \cos\theta$ and $s = \sin\theta$. This matrix, when applied as a similarity transformation to a symmetric matrix A, rotates rows and columns p and q of A through the angle θ so that the (p, q) and (q, p) entries are zeroed. We call the matrix $J(p, q, \theta)$ a *Jacobi rotation*. It is actually identical to a Givens rotation, but in this context we call it a Jacobi rotation to acknowledge its inventor.

Let $\text{off}(A)$ be the square root of the sum of squares of all off-diagonal elements of A. That is,

$$\text{off}(A)^2 = \|A\|_F^2 - \sum_{i=1}^{n} a_{ii}^2.$$

Furthermore, let

$$B = J(p, q, \theta)^T A J(p, q, \theta).$$

Then, because the Frobenius norm is invariant under orthogonal transformations, and because only rows and columns p and q of A are modified in B, we have

$$\begin{aligned}
\text{off}(B)^2 &= \|B\|_F^2 - \sum_{i=1}^{n} b_{ii}^2 \\
&= \|A\|_F^2 - \sum_{i \neq p,q} b_{ii}^2 - (b_{pp}^2 + b_{qq}^2) \\
&= \|A\|_F^2 - \sum_{i \neq p,q} a_{ii}^2 - (a_{pp}^2 + 2a_{pq}^2 + a_{qq}^2) \\
&= \|A\|_F^2 - \sum_{i=1}^{n} a_{ii}^2 - 2a_{pq}^2 \\
&= \text{off}(A)^2 - 2a_{pq}^2 \\
&< \text{off}(A)^2.
\end{aligned}$$

We see that the "size" of the off-diagonal part of the matrix is guaranteeed to decrease from such a similarity transformation.

The 2-by-2 Symmetric Schur Decomposition

We now determine the values c and s such that the diagonalization

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T \begin{bmatrix} a_{pp} & a_{pq} \\ a_{pq} & a_{qq} \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix} = \begin{bmatrix} b_{pp} & 0 \\ 0 & b_{qq} \end{bmatrix}$$

is achieved. From the above matrix equation, we obtain the equation

$$0 = b_{pq} = a_{pq}(c^2 - s^2) + (a_{pp} - a_{qq})cs.$$

If we define

$$\tau = \frac{a_{qq} - a_{pp}}{2a_{pq}}, \quad t = \frac{s}{c},$$

then t satisfies the quadratic equation

$$t^2 + 2\tau t - 1 = 0.$$

Choosing the root that is smaller in absolute value,

$$t = -\tau + / - \sqrt{1 + \tau^2} = \tan\theta,$$

and using the relationships

$$\frac{s}{c} = t, \quad s^2 + c^2 = 1,$$

we obtain

$$c = \frac{1}{sqrt1 + t^2}, \quad s = ct.$$

We choose the smaller root to ensure that $|\theta| \leq \pi/4$ and that the difference between A and B, since

$$\|B - A\|_F^2 = 4(1 - c) \sum_{i \neq p,q} (a_{ip}^2 + a_{iq})^2 + 2a_p q^2/c^2.$$

Therefore, we want c to be larger, which is accomplished if t is smaller.

The Classical Jacobi Algorithm

The *classical Jacobi algorithm* proceeds as follows: find indices p and q, $p \neq q$, such that $|a_{pq}|$ is maximized. Then, we use a single Jacobi rotation to zero a_{pq}, and then repeat this process until $\text{off}(A)$ is sufficiently small.

Let $N = n(n-1)/2$. A sequence of N Jacobi rotations is called a *sweep*. Because, during each iteration, a_{pq} is the largest off-diagonal entry, it follows that

$$\text{off}(A)^2 \leq 2N a_{pq}^2.$$

Because $\text{off}(B)^2 = \text{off}(A)^2 - 2a_{pq}^2$, we have

$$\text{off}(B)^2 \leq \left(1 - \frac{1}{N}\right) \text{off}(A)^2,$$

which implies linear convergence. However, it has been shown that for sufficiently large k, there exist a constant c such that

$$\text{off}(A^{(k+N)}) \leq c * \text{off}(A^{(k)})^2,$$

where $A^{(k)}$ is the matrix after k Jacobi updates, meaning that the clasical Jacobi algorithm converges quadratically as a function of sweeps. Heuristically, it has been argued that approximatly $\log n$ sweeps are needed in practice.

It is worth noting that the guideline that θ be chosen so that $|\theta| \leq \pi/4$ is actually essential to ensure quadratic convergence, because otherwise it is possible that Jacobi updates may simply interchange nearly converged diagonal entries.

The Cyclic-by-Row Algorithm

The classical Jacobi algorithm is impractical because it requires $O(n^2)$ comparisons to find the largest off-diagonal element. A variation, called the *cyclic-by-row* algorithm, avoids this expense by simply cycling through the rows of A, in order. It can be shown that quadratic convergence is still achieved.

Error Analysis

In terms of floating-point operations and comparisons, the Jacobi method is not competitive with the symmetric QR algorithm, as the expense of two Jacobi sweeps is comparable to that of the entire symmetric QR algorithm, even with the accumulation of transformations to obtain the matrix of eigenvectors. On the other hand, the Jacobi method can exploit a known approximate eigenvector matrix, whereas the symmetric QR algorithm cannot.

The relative error in the computed eigenvalues is quite small if A is positive definite. If λ_i is an exact eigenvalue of A and $\tilde{\lambda}_i$ is the closest computed eigenvalue, then it has been shown by Demmel and Veselič that

$$\frac{|\tilde{\lambda}_i - \lambda_i|}{|\lambda_i|} \approx \mathbf{u}\kappa_2(D^{-1}AD^{-1}) \ll \mathbf{u}\kappa_2(A),$$

where D is a diagonal matrix with diagonal entries $\sqrt{a_{11}}, \sqrt{a_{22}}, \ldots, \sqrt{a_{nn}}$.

Parallel Jacobi

The primary advantage of the Jacobi method over the symmetric QR algorithm is its parallelism. As each Jacobi update consists of a row rotation that affects only rows p and q, and a column rotation that effects only columns p and q, up to $n/2$ Jacobi updates can be performed in parallel. Therefore, a sweep can be efficiently implemented by performing $n-1$ series of $n/2$ parallel updates in which each row i is paired with a different row j, for $i \neq j$.

As the size of the matrix, n, is generally much greater than the number of processors, p, it is common to use a *block* approach, in which each update consists of the computation of a $2r \times 2r$ symmetric Schur decomposition for some chosen block size r. This is accomplished by applying another algorithm, such as the symmetric QR algorithm, on a smaller scale. Then, if $p \geq n/(2r)$, an entire block Jacobi sweep can be parallelized.

Jacobi SVD Procedures

The Jacobi method can be adapted to compute the SVD, just as the symmetric QR algorithm is. Two types of Jacobi SVD procedures are:

- *Two-sided Jacobi*: In each Jacobi update, a 2×2 SVD is computed in place of a 2×2 Schur decomposition, using a pair of rotations to zero out the off-diagonal entries a_{pq} and a_{qp}. This process continues until $\text{off}(A)$, whose square is is reduced by $a_{pq}^2 + a_{qp}^2$, is sufficiently small. The idea is to first use a row rotation to make the block symmetric, and then perform a Jacobi update as in the symmetric eigenvalue problem to diagonalize the symmetrized block.

- *One-sided Jacobi:* This approach, like the Golub-Kahan SVD algorithm, implicitly applies the Jacobi method for the symmetric eigenvalue problem to A^TA. The idea is, within each update, to use a column Jacobi rotation to rotate columns p and q of A so that they are

orthogonal, which has the effect of zeroing the (p, q) entry of $A^T A$. Once all columns of AV are orthogonal, where V is the accumulation of all column rotations, the relation $AV = U\Sigma$ is used to obtain U and Σ by simple column scaling. To find a suitable rotation, we note that if $\mathbf{a}_p$ and $\mathbf{a}_q$, the pth and qth columns of A, are rotated through an angle θ, then the rotated columns satisfy

$$(c\mathbf{a}_p - s\mathbf{a}_q)^T(s\mathbf{a}_p + c\mathbf{a}_q) = cs(\|\mathbf{a}_p\|^2 - \|\mathbf{a}_q\|^2) + 2(c^2 - s^2)\mathbf{y}^T\mathbf{x},$$

where $c = \cos\theta$ and $s = \sin\theta$. Dividing by c^2 and defining $t = s/c$, we obtain a quadratic equation for t that can be solved to obtain c and s.