Notes on the Symmetric QR Algorithm

Robert A. van de Geijn
Department of Computer Science
The University of Texas
Austin, TX 78712
rvdg@cs.utexas.edu

November 4, 2014

The QR algorithm is a standard method for computing all eigenvalues and eigenvectors of a matrix. In this note, we focus on the real valued symmetric eigenvalue problem (the case where $A \in \mathbb{R}^{n \times n}$. For this case, recall the Spectral Decomposition Theorem:

Theorem 1. *If $A \in \mathbb{R}^{n \times n}$ then there exists unitary matrix Q and diagonal matrix Λ such that $A = Q \Lambda Q^T$.*

We will partition $Q = \left(\begin{array}{c|c|c} q_0 & \cdots & q_{n-1} \end{array} \right)$ and assume that $\Lambda = \mathrm{diag}(\lambda_0, \cdots, \lambda_{n-1})$ so that throughout this note, q_i and λ_i refer to the ith column of Q and the ith diagonal element of Λ, which means that each tuple (λ, q_i) is an eigenpair.

1 Subspace Iteration

We start with a matrix $V \in \mathbb{R}^{n \times r}$ with normalized columns and iterate something like

$V^{(0)} = V$
for $k = 0, \ldots$ convergence
  $V^{(k+1)} = A V^{(k)}$
  Normalize the columns to be of unit length.
end for

The problem with this approach is that all columns will (likely) converge to an eigenvector associated with the dominant eigenvalue, since the Power Method is being applied to all columns simultaneously. We will now lead the reader through a succession of insights towards a practical algorithm.

Let us examine what $\widehat{V} = AV$ looks like, for the simple case where $V = \left(\begin{array}{c|c|c} v_0 & v_1 & v_2 \end{array} \right)$ (three columns). We know that

$$v_j = Q \underbrace{Q^T v_j}_{y_j}.$$

Hence

$$v_0 = \sum_{j=0}^{n-1} \psi_{0,j} q_j,$$

$$
\begin{array}{rcl}
v_1 & = & \sum_{j=0}^{n-1} \psi_{1,j} q_j, and \\
v_2 & = & \sum_{j=0}^{n-1} \psi_{2,j} q_j,
\end{array}
$$

where $\psi_{i,j}$ equals the ith element of y_j. Then

$$
\begin{array}{rcl}
AV & = & A\left(\begin{array}{c|c|c} v_0 & v_1 & v_2 \end{array} \right) \\
& = & A\left(\begin{array}{c|c|c} \sum_{j=0}^{n-1} \psi_{0,j} q_j & \sum_{j=0}^{n-1} \psi_{1,j} q_j & \sum_{j=0}^{n-1} \psi_{2,j} q_j \end{array} \right) \\
& = & \left(\begin{array}{c|c|c} \sum_{j=0}^{n-1} \psi_{0,j} A q_j & \sum_{j=0}^{n-1} \psi_{1,j} A q_j & \sum_{j=0}^{n-1} \psi_{2,j} A q_j \end{array} \right) \\
& = & \left(\begin{array}{c|c|c} \sum_{j=0}^{n-1} \psi_{0,j} \lambda_j q_j & \sum_{j=0}^{n-1} \psi_{1,j} \lambda_j q_j & \sum_{j=0}^{n-1} \psi_{2,j} \lambda_j q_j \end{array} \right)
\end{array}
$$

- *If* we happened to know λ_0, λ_1, and λ_2 then we could divide the columns by these, respectively, and get new vectors

$$
\begin{array}{rcl}
\left(\begin{array}{c|c|c} \widehat{v}_0 & \widehat{v}_1 & \widehat{v}_2 \end{array} \right) & = & \left(\begin{array}{c|c|c} \sum_{j=0}^{n-1} \psi_{0,j} \left(\frac{\lambda_j}{\lambda_0} \right) q_j & \sum_{j=0}^{n-1} \psi_{1,j} \left(\frac{\lambda_j}{\lambda_1} \right) q_j & \sum_{j=0}^{n-1} \psi_{2,j} \left(\frac{\lambda_j}{\lambda_2} \right) q_j \end{array} \right) \\
& = & \left(\begin{array}{c|c|c} & \psi_{1,0} \left(\frac{\lambda_0}{\lambda_1} \right) q_0 + & \psi_{2,0} \left(\frac{\lambda_0}{\lambda_2} \right) q_0 + \psi_{2,1} \left(\frac{\lambda_1}{\lambda_2} \right) q_1 + \\ \psi_{0,0} q_0 + & \psi_{1,1} q_1 + & \psi_{2,2} q_2 + \\ \sum_{j=1}^{n-1} \psi_{0,j} \left(\frac{\lambda_j}{\lambda_0} \right) q_j & \sum_{j=2}^{n-1} \psi_{1,j} \left(\frac{\lambda_j}{\lambda_1} \right) q_j & \sum_{j=3}^{n-1} \psi_{2,j} \left(\frac{\lambda_j}{\lambda_2} \right) q_j \end{array} \right)
\end{array} \quad (1)
$$

- Assume that $|\lambda_0| > |\lambda_1| > |\lambda_2| > |\lambda_3| \leq \cdots \leq |\lambda_{n-1}|$. Then, similar as for the power method,
  - The first column will see components in the direction of $\{q_1, \ldots, q_{n-1}\}$ shrink relative to the component in the direction of q_0.
  - The second column will see components in the direction of $\{q_2, \ldots, q_{n-1}\}$ shrink relative to the component in the direction of q_1, but the component in the direction of q_0 increases, relatively, since $|\lambda_0/\lambda_1| > 1$.
  - The third column will see components in the direction of $\{q_3, \ldots, q_{n-1}\}$ shrink relative to the component in the direction of q_2, but the components in the directions of q_0 and q_1 increase, relatively, since $|\lambda_0/\lambda_2| > 1$ and $|\lambda_1/\lambda_2| > 1$.

How can we make it so that v_j converges to a vector in the direction of q_j?

- *If* we happen to know q_0, then we can subtract out the component of

$$
\widehat{v}_1 = \psi_{1,0} \frac{\lambda_0}{\lambda_1} q_0 + \psi_{1,1} q_1 + \sum_{j=2}^{n-1} \psi_{1,j} \frac{\lambda_j}{\lambda_1} q_j
$$

  in the direction of q_0:

$$
\widehat{v}_1 - q_0^T \widehat{v}_1 q_0 = \psi_{1,1} q_1 + \sum_{j=2}^{n-1} \psi_{1,j} \frac{\lambda_j}{\lambda_1} q_j
$$

  so that we are left with the component in the direction of q_1 and components in directions of $q_2, \ldots, q_{n-1}$ that are suppressed every time through the loop.

- Similarly, if we also know q_1, the components of $\widehat{v}_2$ in the direction of q_0 and q_1 can be subtracted from that vector.

- We do not know λ_0, λ_1, and λ_2 but from the discussion about the Power Method we remember that we can just normalize the so updated $\widehat{v}_0$, $\widehat{v}_1$, and $\widehat{v}_2$ to have unit length.

How can we make these insights practical?

- We do not know q_0, q_1, and q_2, but we can informally argue that if we keep iterating,
  - The vector $\widehat{v}_0$, normalized in each step, will eventually point in the direction of q_0.
  - $\text{Span}(\widehat{v}_0, \widehat{v}_1)$ will eventually equal $\text{Span}(q_0, q_1)$.
  - In each iteration, we can subtract the component of $\widehat{v}_1$ in the direction of $\widehat{v}_0$ from $\widehat{v}_1$, and then normalize $\widehat{v}_1$ so that eventually result in a the vector that points in the direction of q_1.
  - $\text{Span}(\widehat{v}_0, \widehat{v}_1, \widehat{v}_2)$ will eventually equal $\text{Span}(q_0, q_1, q_2)$.
  - In each iteration, we can subtract the component of $\widehat{v}_2$ in the directions of $\widehat{v}_0$ and $\widehat{v}_1$ from $\widehat{v}_2$, and then normalize the result, to make $\widehat{v}_2$ eventually point in the direction of q_2.

What we recognize is that normalizing $\widehat{v}_0$, subtracting out the component of $\widehat{v}_1$ in the direction of $\widehat{v}_0$, and then normalizing $\widehat{v}_1$, etc., is exactly what the Gram-Schmidt process does. And thus, we can use any convenient (and stable) QR factorization method. This also shows how the method can be generalized to work with more than three columns and even all columns simultaneously.

The algorithm now becomes:

$V^{(0)} = I^{n\times p}$ ($I^{n\times p}$ represents the first p columns of I)
for $k = 0, \ldots$ convergence
  $AV^{(k)} \rightarrow V^{(k+1)}R^{(k+1)}$ (QR factorization with $R^{(k+1)} \in \mathbb{R}^{p\times p}$)
end for

Now consider again (1), focusing on the third column:

$$\begin{pmatrix} \psi_{2,0}\left(\frac{\lambda_0}{\lambda_2}\right) q_0 + \psi_{2,1}\left(\frac{\lambda_1}{\lambda_2}\right) q_1 + \\ \psi_{2,2} q_2 + \\ \sum_{j=3}^{n-1} \psi_j \left(\frac{\lambda_j}{\lambda_2}\right) q_j \end{pmatrix} = \begin{pmatrix} \psi_{2,0}\left(\frac{\lambda_0}{\lambda_2}\right) q_0 + \psi_{2,1}\left(\frac{\lambda_1}{\lambda_2}\right) q_1 + \\ \psi_{2,2} q_2 + \\ \psi_j \boxed{\left(\frac{\lambda_3}{\lambda_2}\right)} q_3 + \sum_{j=4}^{n-1} \psi_{2,j}\left(\frac{\lambda_j}{\lambda_2}\right) q_j \end{pmatrix}.$$

This shows that, if the components in the direction of q_0 and q_1 are subtracted out, it is the component in the direction of q_3 that is deminished in length the most slowly, dictated by the ratio $\left|\frac{\lambda_3}{\lambda_2}\right|$. This, of course, generalizes: the jth column of $V^{(k)}$, $v_i^{(k)}$ will have a component in the direction of q_{j+1}, of length $|q_{j+1}^T v_j^{(k)}|$, that can be expected to shrink most slowly.

We demonstrate this in Figure 1, which shows the execution of the algorithm with $p = n$ for a 5×5 matrix, and shows how $|q_{j+1}^T v_j^{(k)}|$ converge to zero as as function of k.

Next, we observe that if $V \in \mathbb{R}^{n\times n}$ in the above iteration (which means we are iterating with n vectors at a time), then AV yields a next-to last column of the form

$$\begin{pmatrix} \sum_{j=0}^{n-3} \gamma_{n-2,j}\left(\frac{\lambda_j}{\lambda_{n-2}}\right) q_j + \\ \psi_{n-2,n-2} q_{n-2} + \\ \psi_{n-2,n-1} \boxed{\left(\frac{\lambda_{n-1}}{\lambda_{n-2}}\right)} q_{n-1} \end{pmatrix},$$

where $\psi_{i,j} = q_j^T v_i$. Thus, given that the components in the direction of q_j, $j = 0, \ldots, n-2$ can be expected in later iterations to be greatly reduced by the QR factorization that subsequently happens with AV, we notice that it is $\left|\frac{\lambda_{n-1}}{\lambda_{n-2}}\right|$ that dictates how fast the component in the direction of q_{n-1} disappears from $v_{n-2}^{(k)}$. This

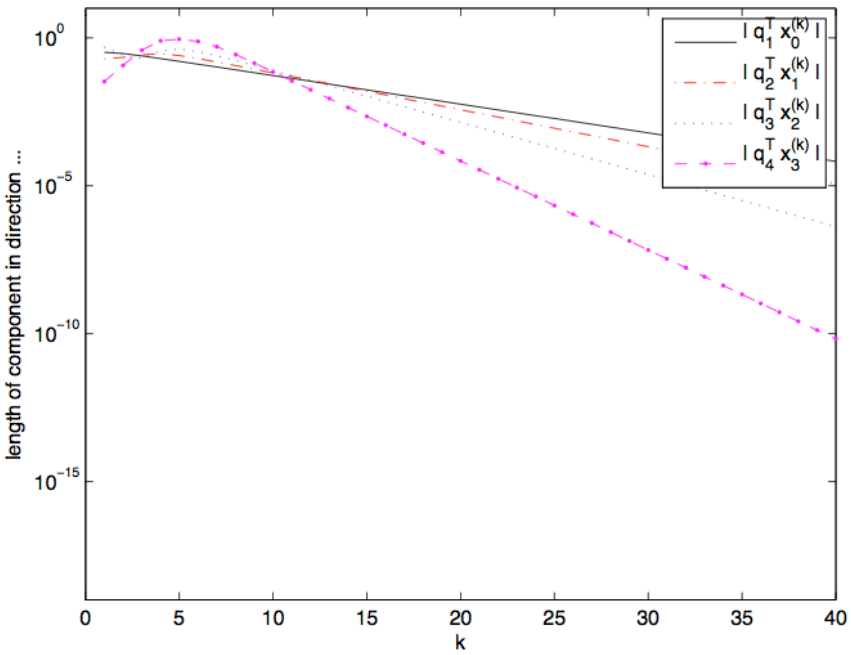

Figure 1: Convergence of the subspace iteration for a 5×5 matrix. This graph is mislabeled: x should be labeled with v. The (linear) convergence of v_j to a vector in the direction of q_j is dictated by now quickly the component in the direction q_{j+1} converges to zero. The line labeled $|q_{j+1}^T x_j|$ plots the length of the component in the direction q_{j+1} as a function of the iteration number.

is a ratio we also saw in the Inverse Power Method and that we noticed we could accelerate in the Rayleigh Quotient Iteration: At each iteration we should shift the matrix to $(A - \mu_k I)$ where $\mu_k \approx \lambda_{n-1}$. Since the last column of $V^{(k)}$ is supposed to be converging to q_{n-1}, it seems reasonable to use $\mu_k = v_{n-1}^{(k)T} A v_{n-1}^{(k)}$ (recall that $v_{n-1}^{(k)}$ has unit length, so this is the Rayleigh quotient.)

The above discussion motivates the iteration

$V^{(0)} := I$ $\quad$ $(V^{(0)} \in \mathbb{R}^{n \times n}!)$

for $k := 0, \ldots$ convergence

$\quad \mu_k := v_{n-1}^{(k)T} A v_{n-1}^{(k)}$ $\quad$ (Rayleigh quotient)

$\quad (A - \mu_k I) V^{(k)} \rightarrow V^{(k+1)} R^{(k+1)}$ $\quad$ (QR factorization)

end for

Notice that this does *not* require one to solve with $(A - \mu_k I)$, unlike in the Rayleigh Quotient Iteration. However, it does require a QR factorization, which requires more computation than the LU factorization (approximately $\frac{4}{3}n^3$ flops).

We demonstrate the convergence in Figure 2, which shows the execution of the algorithm with a 5×5 matrix and illustrates how $|q_j^T v_{n-1}^{(k)}|$ converge to zero as as function of k.

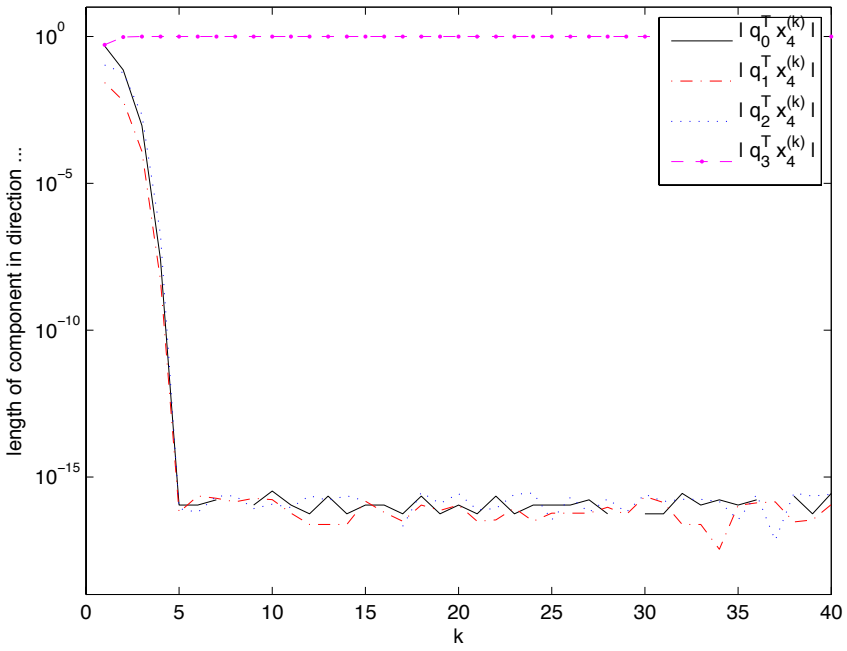

Figure 2: Convergence of the shifted subspace iteration for a 5×5 matrix. This graph is mislabeled: x should be labeled with v. What this graph shows is that the components of v_4 in the directions q_0 throught q_3 disappear very quickly. The vector v_4 quickly points in the direction of the eigenvector associated with the smallest (in magnitude) eigenvalue. Just like the Rayleigh-quotient iteration is not guaranteed to converge to the eigenvector associated with the smallest (in magnitude) eigenvalue, the shifted subspace iteration may home in on a different eigenvector than the one associated with the smallest (in magnitude) eigenvalue. **Something is wrong in this graph: All curves should quickly drop to (near) zero!**

2 The QR Algorithm

The QR algorithm is a classic algorithm for computing all eigenvalues and eigenvectors of a matrix. While we explain it for the symmetric eigenvalue problem, it generalizes to the nonsymmetric eigenvalue problem as well.

2.1 A basic (unshifted) QR algorithm

We have informally argued that the columns of the orthogonal matrices $V^{(k)} \in \mathbb{R}^{n \times n}$ generated by the (unshifted) subspace iteration converge to eigenvectors of matrix A. (The exact conditions under which this happens have not been fully discussed.) In Figure 3 (left), we restate the subspace iteration. In it, we denote matrices $V^{(k)}$ and $R^{(k)}$ from the subspace iteration by $\widehat{V}^{(k)}$ and $\widehat{R}$ to distinguish them from the ones computed by the algorithm on the right. The algorithm on the left also computes the matrix $\widehat{A}^{(k)} = V^{(k)T} A V^{(k)}$, a matrix that hopefully converges to Λ, the diagonal matrix with the eigenvalues of A on its diagonal. To the right is the QR algorithm. The claim is that the two algorithms compute the same quantities.

Exercise 2. *Prove that in Figure 3,* $\widehat{V}^{(k)} = V^{(k)}$, *and* $\widehat{A}^{(k)} = A^{(k)}$, $k = 1, \ldots$.

Subspace iteration	QR algorithm
$\widehat{A}^{(0)} := A$	$A^{(0)} := A$
$\widehat{V}^{(0)} := I$	$V^{(0)} := I$
for $k := 0, \ldots$ until convergence	for $k := 0, \ldots$ until convergence
$A\widehat{V}^{(k)} \rightarrow \widehat{V}^{(k+1)}\widehat{R}^{(k+1)}$ (QR factorization)	$A^{(k)} \rightarrow Q^{(k+1)}R^{(k+1)}$ (QR factorization)
$\widehat{A}^{(k+1)} := \widehat{V}^{(k+1)T}A\widehat{V}^{(k+1)}$	$A^{(k+1)} := R^{(k+1)}Q^{(k+1)}$
	$V^{(k+1)} := V^{(k)}Q^{(k+1)}$
end for	end for

Figure 3: Basic subspace iteration and basic QR algorithm.

Subspace iteration	QR algorithm
$\widehat{A}^{(0)} := A$	$A^{(0)} := A$
$\widehat{V}^{(0)} := I$	$V^{(0)} := I$
for $k := 0, \ldots$ until convergence	for $k := 0, \ldots$ until convergence
$\widehat{\mu}_k := v_{n-1}^{(k)\,T} A v_{n-1}^{(k)}$	$\mu_k = \alpha_{n-1,n-1}^{(k)}$
$(A - \widehat{\mu}I)\widehat{V}^{(k)} \rightarrow \widehat{V}^{(k+1)}\widehat{R}^{(k+1)}$ (QR factorization)	$A^{(k)} - \mu_k I \rightarrow Q^{(k+1)}R^{(k+1)}$ (QR factorization)
$\widehat{A}^{(k+1)} := \widehat{V}^{(k+1)T}A\widehat{V}^{(k+1)}$	$A^{(k+1)} := R^{(k+1)}Q^{(k+1)} + \mu_k I$
	$V^{(k+1)} := V^{(k)}Q^{(k+1)}$
end for	end for

Figure 4: Basic shifted subspace iteration and basic shifted QR algorithm.

We conclude that if $\widehat{V}^{(k)}$ converges to the matrix of orthonormal eigenvectors when the subspace iteration is applied to $V^{(0)} = I$, then $A^{(k)}$ converges to the diagonal matrix with eigenvalues along the diagonal.

2.2 A basic shifted QR algorithm

In Figure 4 (left), we restate the subspace iteration with shifting. In it, we denote matrices $V^{(k)}$ and $R^{(k)}$ from the subspace iteration by $\widehat{V}^{(k)}$ and $\widehat{R}$ to distinguish them from the ones computed by the algorithm on the right. The algorithm on the left also computes the matrix $\widehat{A}^{(k)} = V^{(k)T}AV^{(k)}$, a matrix that hopefully converges to Λ, the diagonal matrix with the eigenvalues of A on its diagonal. To the right is the shifted QR algorithm. The claim is that the two algorithms compute the same quantities.

Exercise 3. *Prove that in Figure 4,* $\widehat{V}^{(k)} = V^{(k)}$, *and* $\widehat{A}^{(k)} = A^{(k)}$, $k = 1, \ldots$.

We conclude that if $\widehat{V}^{(k)}$ converges to the matrix of orthonormal eigenvectors when the shifted subspace iteration is applied to $V^{(0)} = I$, then $A^{(k)}$ converges to the diagonal matrix with eigenvalues along the diagonal.

The convergence of the basic shifted QR algorithm is illustrated below. Pay particular attention to the convergence of the last row and column.

$$A^{(0)} = \begin{pmatrix} 2.01131953448 & 0.05992695085 & 0.14820940917 \\ 0.05992695085 & 2.30708673171 & 0.93623515213 \\ 0.14820940917 & 0.93623515213 & 1.68159373379 \end{pmatrix} \quad A^{(1)} = \begin{pmatrix} 2.21466116574 & 0.34213192482 & 0.31816754245 \\ 0.34213192482 & 2.54202325042 & 0.57052186467 \\ 0.31816754245 & 0.57052186467 & 1.24331558383 \end{pmatrix}$$

$$A^{(2)} = \begin{pmatrix} 2.63492207667 & 0.47798481637 & 0.07654607908 \\ 0.47798481637 & 2.35970859985 & 0.06905042811 \\ 0.07654607908 & 0.06905042811 & 1.00536932347 \end{pmatrix} \quad A^{(3)} = \begin{pmatrix} 2.87588550968 & 0.32971207176 & 0.00024210487 \\ 0.32971207176 & 2.12411444949 & 0.00014361630 \\ 0.00024210487 & 0.00014361630 & 1.00000004082 \end{pmatrix}$$

$$A^{(4)} = \begin{pmatrix} 2.96578660126 & 0.18177690194 & 0.00000000000 \\ 0.18177690194 & 2.03421339873 & 0.00000000000 \\ 0.00000000000 & 0.00000000000 & 1.00000000000 \end{pmatrix} \quad A^{(5)} = \begin{pmatrix} 2.9912213907 & 0.093282073553 & 0.00000000000 \\ 0.0932820735 & 2.008778609226 & 0.00000000000 \\ 0.0000000000 & 0.000000000000 & 1.00000000000 \end{pmatrix}$$

Once the off-diagonal elements of the last row and column have converged (are sufficiently small), the problem can be *deflated* by applying the following theorem:

Theorem 4. *Let*

$$A = \left(\begin{array}{c|c|c|c} A_{0,0} & A_{01} & \cdots & A_{0N-1} \\ \hline 0 & A_{1,1} & \cdots & A_{1,N-1} \\ \hline \vdots & \vdots & \ddots & A_{00} \\ \hline 0 & 0 & \cdots & A_{N-1,N-1} \end{array}\right),$$

where $A_{k,k}$ are all square. Then $\lambda(A) = \cup_{k=0}^{N-1} \lambda(A_{k,k})$.

Exercise 5. *Prove the above theorem.*

In other words, once the last row and column have converged, the algorithm can continue with the submatrix that consists of the first $n-1$ rows and columns.

The problem with the QR algorithm, as stated, is that each iteration requires $O(n^3)$ operations, which is too expensive given that many iterations are required to find all eigenvalues and eigenvectors.

3 Reduction to Tridiagonal Form

In the next section, we will see that if $A^{(0)}$ is a tridiagonal matrix, then so are all $A^{(k)}$. This reduces the cost of each iteration from $O(n^3)$ to $O(n)$. We first show how unitary similarity transformations can be used to reduce a matrix to tridiagonal form.

3.1 Householder transformations (reflectors)

We briefly review the main tool employed to reduce a matrix to tridiagonal form: the Householder transform, also known as a reflector. Full details were given in "Notes on Householder QR Factorization".

Definition 6. *Let $u \in \mathbb{R}^n$, $\tau \in \mathbb{R}$. Then $H = H(u) = I - uu^T/\tau$, where $\tau = \frac{1}{2}u^T u$, is said to be a reflector or Householder transformation.*

We observe:

- Let z be any vector that is perpendicular to u. Applying a Householder transform $H(u)$ to z leaves the vector unchanged: $H(u)z = z$.
- Let any vector x be written as $x = z + u^T x u$, where z is perpendicular to u and $u^T x u$ is the component of x in the direction of u. Then $H(u)x = z - u^T x u$.

This can be interpreted as follows: The space perpendicular to u acts as a "mirror": any vector in that space (along the mirror) is not reflected, while any other vector has the component that is orthogonal to the space (the component outside and orthogonal to the mirror) reversed in direction. Notice that a reflection preserves the length of the vector. Also, it is easy to verify that:

1. $HH = I$ (reflecting the reflection of a vector results in the original vector);
2. $H = H^T$, and so $H^T H = HH^T = I$ (a reflection is an orthogonal matrix and thus preserves the norm); and
3. if $H_0, \cdots, H_{k-1}$ are Householder transformations and $Q = H_0 H_1 \cdots H_{k-1}$, then $Q^T Q = QQ^T = I$ (an accumulation of reflectors is an orthogonal matrix).

As part of the reduction to condensed form operations, given a vector x we will wish to find a Householder transformation, $H(u)$, such that $H(u)x$ equals a vector with zeroes below the first element: $H(u)x = \mp\|x\|_2 e_0$ where e_0 equals the first column of the identity matrix. It can be easily checked that choosing $u = x \pm \|x\|_2 e_0$ yields the desired $H(u)$. Notice that any nonzero scaling of u has the same property, and the convention is to scale u so that the first element equals one. Let us define $[u, \tau, h] = \text{Hous}(x)$ to be the function that returns u with first element equal to one, $\tau = \frac{1}{2}u^T u$, and $h = H(u)x$.

3.2 Algorithm

The first step towards computing the eigenvalue decomposition of a symmetric matrix is to reduce the matrix to tridiagonal form.

The basic algorithm for reducing a symmetric matrix to tridiagonal form, overwriting the original matrix with the result, can be explained as follows. We assume that symmetric A is stored only in the lower triangular part of the matrix and that only the diagonal and subdiagonal of the symmetric tridiagonal matrix is computed, overwriting those parts of A. Finally, the Householder vectors used to zero out parts of A overwrite the entries that they annihilate (set to zero).

- Partition $A \rightarrow \left(\begin{array}{c|c} \alpha_{11} & a_{21}^T \\ \hline a_{21} & A_{22} \end{array}\right)$.
- Let $[u_{21}, \tau, a_{21}] := \text{Hous}(a_{21})$.[1]
- Update

$$\left(\begin{array}{c|c} \alpha_{11} & a_{21}^T \\ \hline a_{21} & A_{22} \end{array}\right) := \left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & H \end{array}\right)\left(\begin{array}{c|c} \alpha_{11} & a_{21}^T \\ \hline a_{21} & A_{22} \end{array}\right)\left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & H \end{array}\right) = \left(\begin{array}{c|c} \alpha_{11} & a_{21}^T H \\ \hline Ha_{21} & HA_{22}H \end{array}\right)$$

  where $H = H(u_{21})$. Note that $a_{21} := Ha_{21}$ need not be executed since this update was performed by the instance of Hous above.[2] Also, a_{12}^T is not stored nor updated due to symmetry. Finally, only the lower triangular part of $HA_{22}H$ is computed, overwriting A_{22}. The update of A_{22} warrants closer scrutiny:

$$\begin{array}{rcl} A_{22} & := & (I - \frac{1}{\tau}u_{21}u_{21}^T)A_{22}(I - \frac{1}{\tau}u_{21}u_{21}^T) \\ & = & (A_{22} - \frac{1}{\tau}u_{21}\underbrace{u_{21}^T A_{22}}_{y_{21}^T})(I - \frac{1}{\tau}u_{21}u_{21}^T) \end{array}$$

[1] Note that the semantics here indicate that a_{21} is overwritten by Ha_{21}.

[2] In practice, the zeros below the first element of Ha_{21} are not actually written. Instead, the implementation overwrites these elements with the corresponding elements of the vector u_{21}.

Algorithm: $[A] := \text{TRIRED\_UNB}(b, A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$

where A_{TL} is 0×0

while $m(A_{TL}) < m(A)$ **do**

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right)$$

where α_{11} is a scalar

$[u_{21}, \tau, a_{21}] := \text{Hous}(a_{21})$

$y_{21} := A_{22} u_{21}$

$\beta := u_{21}^T y_{21} / 2$

$w_{21} := (y_{21} - \beta u_{21}/\tau)/\tau$

$A_{22} := A_{22} - \text{TRIL}(u_{21} w_{21}^T + w_{21} u_{21}^T)$ (symmetric rank-2 update)

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right)$$

endwhile

Figure 5: Basic algorithm for reduction of a symmetric matrix to tridiagonal form.

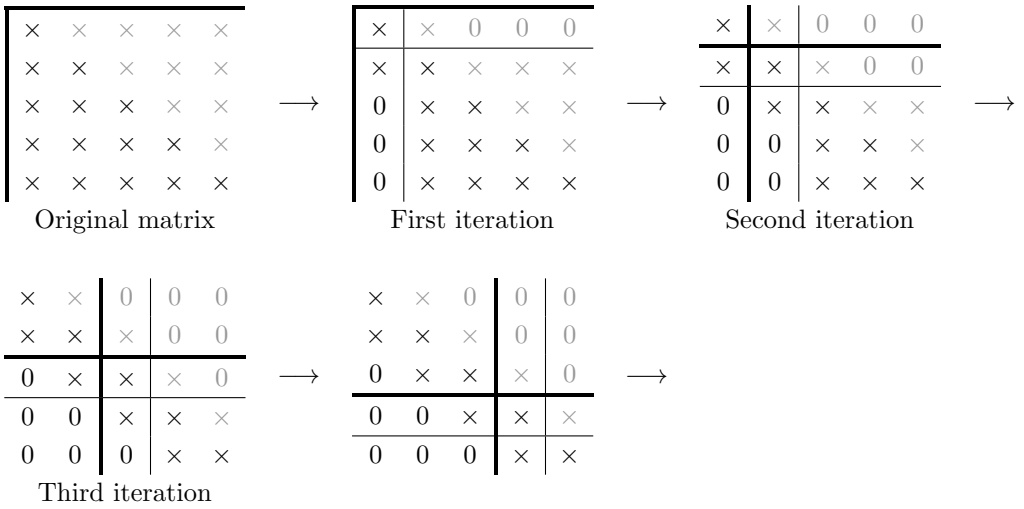

Figure 6: Illustration of reduction of a symmetric matrix to tridiagonal form. The ×s denote nonzero elements in the matrix. The gray entries above the diagonal are not actually updated.

$$
\begin{array}{rcl}
 & = & A_{22} - \frac{1}{\tau} u_{21} y_{21}^T - \frac{1}{\tau} \underbrace{A u_{21}}_{y_{21}} u_{21}^T + \frac{1}{\tau^2} u_{21} \underbrace{y_{21}^T u_{21}}_{2\beta} u_{21}^T \\
 & = & A_{22} - \left(\frac{1}{\tau} u_{21} y_{21}^T + \frac{\beta}{\tau^2} u_{21} u_{21}^T \right) - \left(\frac{1}{\tau} y_{21} u_{21}^T + \frac{\beta}{\tau^2} u_{21} u_{21}^T \right) \\
 & = & A_{22} - u_{21} \underbrace{\frac{1}{\tau} \left(y_{21}^T + \frac{\beta}{\tau} u_{21}^T \right)}_{w_{21}^T} - \underbrace{\frac{1}{\tau} \left(y_{21} + \frac{\beta}{\tau} u_{21} \right)}_{w_{21}} u_{21}^T \\
 & = & \underbrace{A_{22} - u_{21} w_{21}^T - w_{21} u_{21}^T}_{\begin{array}{c}\text{symmetric} \\ \text{rank-2 update}\end{array}}.
\end{array}
$$

- Continue this process with the updated A_{22}.

This is captured in the algorithm in Figure 5. It is also illustrated in Figure 6.

The total cost for reducing $A \in \mathbb{R}^{n \times n}$ is approximately

$$
\sum_{k=0}^{n-1} \left(4(n-k-1)^2 \right) \text{ flops} \approx \frac{4}{3} n^3 \text{ flops}.
$$

This equals, approximately, the cost of one QR factorization of matrix A.

4 The QR algorithm with a Tridiagonal Matrix

We are now ready to describe an algorithm for the QR algorithm with a tridiagonal matrix.

4.1 Givens' rotations

First, we introduce another important class of unitary matrices known as Givens' rotations. Given a vector $x = \begin{pmatrix} \chi_1 \\ \chi_2 \end{pmatrix} \in \mathbb{R}^2$, there exists an orthogonal matrix G such that $G^T x = \begin{pmatrix} \pm \|x\|_2 \\ 0 \end{pmatrix}$. The Householder transformation is one example of such a matrix G. An alternative is the Givens' rotation: $G = \begin{pmatrix} \gamma & -\sigma \\ \sigma & \gamma \end{pmatrix}$ where $\gamma^2 + \sigma^2 = 1$. (Notice that γ and σ can be thought of as the cosine and sine of an angle.) Then

$$
\begin{array}{rcl}
G^T G & = & \begin{pmatrix} \gamma & -\sigma \\ \sigma & \gamma \end{pmatrix}^T \begin{pmatrix} \gamma & -\sigma \\ \sigma & \gamma \end{pmatrix} = \begin{pmatrix} \gamma & \sigma \\ -\sigma & \gamma \end{pmatrix} \begin{pmatrix} \gamma & -\sigma \\ \sigma & \gamma \end{pmatrix} \\
 & = & \begin{pmatrix} \gamma^2 + \sigma^2 & -\gamma\sigma + \gamma\sigma \\ \gamma\sigma - \gamma\sigma & \gamma^2 + \sigma^2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},
\end{array}
$$

which means that a Givens' rotation is a unitary matrix.

Now, if $\gamma = \chi_1 / \|x\|_2$ and $\sigma = \chi_2 / \|x\|_2$, then $\gamma^2 + \sigma^2 = (\chi_1^2 + \chi_2^2)/\|x\|_2^2 = 1$ and

$$
\begin{pmatrix} \gamma & -\sigma \\ \sigma & \gamma \end{pmatrix}^T \begin{pmatrix} \chi_1 \\ \chi_2 \end{pmatrix} = \begin{pmatrix} \gamma & \sigma \\ -\sigma & \gamma \end{pmatrix} \begin{pmatrix} \chi_1 \\ \chi_2 \end{pmatrix} = \begin{pmatrix} (\chi_1^2 + \chi_2^2)/\|x\|_2 \\ (\chi_1\chi_2 - \chi_1\chi_2)/\|x\|_2 \end{pmatrix} = \begin{pmatrix} \|x\|_2 \\ 0 \end{pmatrix}.
$$

5 QR Factorization of a Tridiagonal Matrix

Now, consider the 4×4 tridiagonal matrix

$$\begin{pmatrix} \alpha_{0,0} & \alpha_{0,1} & 0 & 0 \\ \alpha_{1,0} & \alpha_{1,1} & \alpha_{1,2} & 0 \\ 0 & \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} \\ 0 & 0 & \alpha_{3,2} & \alpha_{3,3} \end{pmatrix}$$

From $\begin{pmatrix} \alpha_{0,0} \\ \alpha_{1,0} \end{pmatrix}$ one can compute $\gamma_{1,0}$ and $\sigma_{1,0}$ so that

$$\begin{pmatrix} \gamma_{1,0} & -\sigma_{1,0} \\ \sigma_{1,0} & \gamma_{1,0} \end{pmatrix}^T \begin{pmatrix} \alpha_{0,0} \\ \alpha_{1,0} \end{pmatrix} = \begin{pmatrix} \widehat{\alpha}_{0,0} \\ 0 \end{pmatrix}.$$

Then

$$\left(\begin{array}{cc|cc} \widehat{\alpha}_{0,0} & \widehat{\alpha}_{0,1} & \widehat{\alpha}_{0,2} & 0 \\ 0 & \widehat{\alpha}_{1,1} & \widehat{\alpha}_{1,2} & 0 \\ \hline 0 & \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} \\ 0 & 0 & \alpha_{3,2} & \alpha_{3,3} \end{array}\right) = \left(\begin{array}{cc|cc} \gamma_{1,0} & \sigma_{1,0} & 0 & 0 \\ -\sigma_{1,0} & \gamma_{1,0} & 0 & 0 \\ \hline 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right) \left(\begin{array}{cc|cc} \alpha_{0,0} & \alpha_{0,1} & 0 & 0 \\ \alpha_{1,0} & \alpha_{1,1} & \alpha_{1,2} & 0 \\ \hline 0 & \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} \\ 0 & 0 & \alpha_{3,2} & \alpha_{3,3} \end{array}\right)$$

Next, from $\begin{pmatrix} \widehat{\alpha}_{1,1} \\ \alpha_{2,1} \end{pmatrix}$ one can compute $\gamma_{2,1}$ and $\sigma_{2,1}$ so that

$$\begin{pmatrix} \gamma_{2,1} & -\sigma_{2,1} \\ \sigma_{2,1} & \gamma_{2,1} \end{pmatrix}^T \begin{pmatrix} \widehat{\alpha}_{1,1} \\ \alpha_{2,1} \end{pmatrix} = \begin{pmatrix} \widehat{\widehat{\alpha}}_{1,1} \\ 0 \end{pmatrix}.$$

Then

$$\left(\begin{array}{c|ccc} \widehat{\alpha}_{0,0} & \widehat{\alpha}_{0,1} & \widehat{\alpha}_{0,2} & 0 \\ \hline 0 & \widehat{\widehat{\alpha}}_{1,1} & \widehat{\widehat{\alpha}}_{1,2} & \widehat{\widehat{\alpha}}_{1,3} \\ 0 & 0 & \widehat{\alpha}_{2,2} & \widehat{\alpha}_{2,3} \\ \hline 0 & 0 & \alpha_{3,2} & \alpha_{3,3} \end{array}\right) = \left(\begin{array}{c|ccc} 1 & 0 & 0 & 0 \\ \hline 0 & \gamma_{2,1} & \sigma_{2,1} & 0 \\ 0 & -\sigma_{2,1} & \gamma_{2,1} & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \left(\begin{array}{c|ccc} \widehat{\alpha}_{0,0} & \widehat{\alpha}_{0,1} & \widehat{\alpha}_{0,2} & 0 \\ \hline 0 & \widehat{\alpha}_{1,1} & \widehat{\alpha}_{1,2} & 0 \\ 0 & \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} \\ \hline 0 & 0 & \alpha_{3,2} & \alpha_{3,3} \end{array}\right)$$

Finally, from $\begin{pmatrix} \widehat{\alpha}_{2,2} \\ \alpha_{3,2} \end{pmatrix}$ one can compute $\gamma_{3,2}$ and $\sigma_{3,2}$ so that $\begin{pmatrix} \gamma_{3,2} & -\sigma_{3,2} \\ \sigma_{3,2} & \gamma_{3,2} \end{pmatrix}^T \begin{pmatrix} \widehat{\alpha}_{2,2} \\ \alpha_{3,2} \end{pmatrix} = \begin{pmatrix} \widehat{\widehat{\alpha}}_{2,2} \\ 0 \end{pmatrix}$.
Then

$$\left(\begin{array}{cc|cc} \widehat{\alpha}_{0,0} & \widehat{\alpha}_{0,1} & \widehat{\alpha}_{0,2} & 0 \\ 0 & \widehat{\widehat{\alpha}}_{1,1} & \widehat{\widehat{\alpha}}_{1,2} & \widehat{\widehat{\alpha}}_{1,3} \\ \hline 0 & 0 & \widehat{\widehat{\alpha}}_{2,2} & \widehat{\widehat{\alpha}}_{2,3} \\ 0 & 0 & 0 & \widehat{\alpha}_{3,3} \end{array}\right) = \left(\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 1 & 0 & \gamma_{3,2} & \sigma_{3,2} \\ 0 & 1 & -\sigma_{3,2} & \gamma_{3,2} \end{array}\right) \left(\begin{array}{cc|cc} \widehat{\alpha}_{0,0} & \widehat{\alpha}_{0,1} & \widehat{\alpha}_{0,2} & 0 \\ 0 & \widehat{\widehat{\alpha}}_{1,1} & \widehat{\widehat{\alpha}}_{1,2} & \widehat{\widehat{\alpha}}_{1,3} \\ \hline 0 & 0 & \widehat{\alpha}_{2,2} & \widehat{\alpha}_{2,3} \\ 0 & 0 & \alpha_{3,2} & \alpha_{3,3} \end{array}\right)$$

The matrix Q is the orthogonal matrix that results from multiplying the different Givens' rotations together:

$$Q = \left(\begin{array}{cc|cc} \gamma_{1,0} & -\sigma_{1,0} & 0 & 0 \\ \sigma_{1,0} & \gamma_{1,0} & 0 & 0 \\ \hline 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right) \left(\begin{array}{c|ccc} 1 & 0 & 0 & 0 \\ \hline 0 & \gamma_{2,1} & -\sigma_{2,1} & 0 \\ 0 & \sigma_{2,1} & \gamma_{2,1} & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \left(\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & \gamma_{3,2} & -\sigma_{3,2} \\ 0 & 0 & \sigma_{3,2} & \gamma_{3,2} \end{array}\right). \tag{2}$$

However, it is typically not explicitly formed.

The next question is how to compute RQ given the QR factorization of the tridiagonal matrix:

$$
\underbrace{\left(\begin{array}{cc|cc}
\widehat{\alpha}_{0,0} & \widehat{\alpha}_{0,1} & \widehat{\alpha}_{0,2} & 0 \\
0 & \widehat{\widehat{\alpha}}_{1,1} & \widehat{\widehat{\alpha}}_{1,2} & \widehat{\widehat{\alpha}}_{1,3} \\
\hline
0 & 0 & \widehat{\widehat{\alpha}}_{2,2} & \widehat{\widehat{\alpha}}_{2,3} \\
0 & 0 & 0 & \widehat{\alpha}_{3,3}
\end{array}\right)
\left(\begin{array}{cc|cc}
\gamma_{1,0} & -\sigma_{1,0} & 0 & 0 \\
\sigma_{1,0} & \gamma_{1,0} & 0 & 0 \\
\hline
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{array}\right)}
\left(\begin{array}{c|cc|c}
1 & 0 & 0 & 0 \\
\hline
0 & \gamma_{2,1} & -\sigma_{2,1} & 0 \\
0 & \sigma_{2,1} & \gamma_{2,1} & 0 \\
\hline
0 & 0 & 0 & 1
\end{array}\right)
\left(\begin{array}{cc|cc}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
\hline
0 & 0 & \gamma_{3,2} & -\sigma_{3,2} \\
0 & 0 & \sigma_{3,2} & \gamma_{3,2}
\end{array}\right)
$$

$$
\underbrace{\left(\begin{array}{cc|cc}
\tilde{\alpha}_{0,0} & \tilde{\alpha}_{0,1} & \widehat{\alpha}_{0,2} & 0 \\
\tilde{\alpha}_{1,0} & \tilde{\widehat{\alpha}}_{1,1} & \widehat{\widehat{\alpha}}_{1,2} & \widehat{\widehat{\alpha}}_{1,3} \\
\hline
0 & 0 & \widehat{\widehat{\alpha}}_{2,2} & \widehat{\widehat{\alpha}}_{2,3} \\
0 & 0 & 0 & \widehat{\alpha}_{3,3}
\end{array}\right)}
\left(\begin{array}{c|ccc}
\tilde{\alpha}_{0,0} & \tilde{\tilde{\alpha}}_{0,1} & \cancel{\tilde{\alpha}_{0,2}} & 0 \\
\hline
\tilde{\alpha}_{1,0} & \tilde{\tilde{\alpha}}_{1,1} & \tilde{\alpha}_{1,2} & \widehat{\widehat{\alpha}}_{1,3} \\
0 & \tilde{\alpha}_{2,1} & \tilde{\alpha}_{2,2} & \widehat{\widehat{\alpha}}_{2,3} \\
\hline
0 & 0 & 0 & \widehat{\alpha}_{3,3}
\end{array}\right)
\underbrace{\left(\begin{array}{c|c|cc}
\tilde{\alpha}_{0,0} & \tilde{\tilde{\alpha}}_{0,1} & \cancel{\tilde{\alpha}_{0,2}} & 0 \\
\hline
\tilde{\alpha}_{1,0} & \tilde{\tilde{\alpha}}_{1,1} & \tilde{\tilde{\alpha}}_{1,2} & \cancel{\tilde{\alpha}_{1,3}} \\
\hline
0 & \tilde{\alpha}_{2,1} & \tilde{\tilde{\alpha}}_{2,2} & \tilde{\alpha}_{2,3} \\
0 & 0 & \tilde{\alpha}_{3,2} & \tilde{\alpha}_{3,3}
\end{array}\right)}
$$

.

A symmetry argument can be used to motivate that $\cancel{\tilde{\alpha}_{0,2}} = \cancel{\tilde{\alpha}_{1,3}} = 0$.

6 The Implictly Shifted QR Algorithm

6.1 Upper Hessenberg matrix

Definition 7. *A matrix is said to be upper Hessenberg if all entries below its first subdiagonal equal zero.*

In other words, if matrix $A \in \mathbb{R}^{n \times n}$ is upper Hessenberg, it looks like

$$
A = \begin{pmatrix}
\alpha_{0,0} & \alpha_{0,1} & \alpha_{0,2} & \cdots & \alpha_{0,n-1} & \alpha_{0,n-1} \\
\alpha_{1,0} & \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,n-1} & \alpha_{1,n-1} \\
0 & \alpha_{2,1} & \alpha_{2,2} & \cdots & \alpha_{2,n-1} & \alpha_{2,n-1} \\
\vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \ddots & \alpha_{n-2,n-2} & \alpha_{n-2,n-2} \\
0 & 0 & 0 & \cdots & \alpha_{n-1,n-2} & \alpha_{n-1,n-2}
\end{pmatrix}.
$$

Obviously, a tridiagonal matrix is a special case of an upper Hessenberg matrix.

6.2 The Implicit Q Theorem

The following theorem sets up one of the most remarkable algorithms in numerical linear algebra, which allows us to greatly simplify the implementation of the shifted QR algorithm when A is tridiagonal.

Theorem 8 (Implicit Q Theorem). *Let $A, B \in \mathbb{R}^{n\times n}$ where B is upper Hessenberg and has only positive elements on its first subdiagonal and assume there exists an orthogonal matrix Q such that $Q^T AQ = B$. Then Q and B are uniquely determined by A and the first column of Q.*

Proof: Notice that $AQ = QB$. Let $Q = \left(\begin{array}{c|c|c} q_0 & q_1 & Q_2 \end{array}\right)$ and $B = \left(\begin{array}{c|c|c} \psi_{00} & \star & \star \\ \hline \psi_{10} & \psi_{11} & \star \\ \hline 0 & \psi_{21}e_0 & B_{22} \end{array}\right)$ and focus on the first column of both sides of $AQ = QB$: $Aq_0 = \left(\begin{array}{c|c} q_0 & q_1 \end{array}\right)\begin{pmatrix} \psi_{00} \\ \psi_{10} \end{pmatrix} = \psi_{00}q_0 + \psi_{10}q_0$. By orthogonality of q_0 and q_1 we find that $\psi_{00} = q_0^T Aq_0$ and $\psi_{10}q_1 = \widehat{q}_1 = Aq_0 - \psi_{00}q_0$. Since $\psi_{10} > 0$ we deduce that $\widehat{q}_1 \neq 0$. Since $\|q_1\|_2 = 1$ we conclude that $\psi_{10} = \|\widehat{q}_1\|_2$ and $q_1 = \widehat{q}_1/\psi_{10}$. The point is that ψ_{00} and ψ_{10} are prescribed, as is q_1. An inductive proof can be constructed to similarly show that the rest of the elements of B and Q are uniquely determined.
□

Notice the similarity between the above proof and the proof of the existence and uniqueness of the QR factorization!

Exercise 9. *Complete the above proof.*

6.3 The Francis QR Step

The Francis QR Step combines the steps $(A^{(k-1)} - \mu_k I) \rightarrow Q^{(k)}R^{(k)}$ and $A^{(k+1)} := R^{(k)}Q^{(k)} + \mu_k I$ into a single step.

Now, consider the 4×4 tridiagonal matrix

$$\begin{pmatrix} \alpha_{0,0} & \alpha_{0,1} & 0 & 0 \\ \alpha_{1,0} & \alpha_{1,1} & \alpha_{1,2} & 0 \\ 0 & \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} \\ 0 & 0 & \alpha_{3,2} & \alpha_{3,3} \end{pmatrix} - \mu I$$

The first Givens' rotation is computed from $\begin{pmatrix} \alpha_{0,0} - \mu \\ \alpha_{1,0} \end{pmatrix}$, yielding $\gamma_{1,0}$ and $\sigma_{1,0}$ so that

$$\begin{pmatrix} \gamma_{1,0} & -\sigma_{1,0} \\ \sigma_{1,0} & \gamma_{1,0} \end{pmatrix}^T \begin{pmatrix} \alpha_{0,0} - \mu I \\ \alpha_{1,0} \end{pmatrix}$$

has a zero second entry. Now, to preserve eigenvalues, any orthogonal matrix that is applied from the left must also have its transpose applied from the right. Let us compute

$$\left(\begin{array}{cc|cc} \tilde{\alpha}_{0,0} & \widehat{\alpha}_{1,0} & \widehat{\alpha}_{2,0} & 0 \\ \widehat{\alpha}_{1,0} & \widehat{\alpha}_{1,1} & \widehat{\alpha}_{1,2} & 0 \\ \hline \widehat{\alpha}_{2,0} & \widehat{\alpha}_{2,1} & \alpha_{2,2} & \alpha_{2,3} \\ \hline 0 & 0 & \alpha_{3,2} & \alpha_{3,3} \end{array}\right) = \left(\begin{array}{cc|cc} \gamma_{1,0} & \sigma_{1,0} & 0 & 0 \\ -\sigma_{1,0} & \gamma_{1,0} & 0 & 0 \\ \hline 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \left(\begin{array}{cc|cc} \alpha_{0,0} & \alpha_{0,1} & 0 & 0 \\ \alpha_{1,0} & \alpha_{1,1} & \alpha_{1,2} & 0 \\ \hline 0 & \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} \\ \hline 0 & 0 & \alpha_{3,2} & \alpha_{3,3} \end{array}\right) \left(\begin{array}{cc|cc} \gamma_{1,0} & -\sigma_{1,0} & 0 & 0 \\ \sigma_{1,0} & \gamma_{1,0} & 0 & 0 \\ \hline 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right).$$

Next, from $\begin{pmatrix} \widehat{\alpha}_{1,0} \\ \widehat{\alpha}_{2,0} \end{pmatrix}$ one can compute $\gamma_{2,0}$ and $\sigma_{2,0}$ so that $\begin{pmatrix} \gamma_{2,0} & -\sigma_{2,0} \\ \sigma_{2,0} & \gamma_{2,0} \end{pmatrix}^T \begin{pmatrix} \widehat{\alpha}_{1,0} \\ \widehat{\alpha}_{2,0} \end{pmatrix} = \begin{pmatrix} \tilde{\alpha}_{1,0} \\ 0 \end{pmatrix}$.

Then

$$\left(\begin{array}{c|ccc} \tilde{\alpha}_{0,0} & \tilde{\alpha}_{1,0} & 0 & 0 \\ \hline \tilde{\alpha}_{1,0} & \tilde{\alpha}_{1,1} & \widehat{\widehat{\alpha}}_{2,1} & \widehat{\alpha}_{3,1} \\ 0 & \widehat{\widehat{\alpha}}_{2,1} & \widehat{\alpha}_{2,2} & \widehat{\alpha}_{2,3} \\ \hline 0 & \widehat{\alpha}_{3,1} & \widehat{\alpha}_{3,2} & \alpha_{3,3} \end{array}\right) = \left(\begin{array}{c|ccc} 1 & 0 & 0 & 0 \\ \hline 0 & \gamma_{2,0} & \sigma_{2,0} & 0 \\ 0 & -\sigma_{2,0} & \gamma_{2,0} & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \left(\begin{array}{c|ccc} \tilde{\alpha}_{0,0} & \widehat{\alpha}_{1,0} & \widehat{\alpha}_{2,0} & 0 \\ \hline \widehat{\alpha}_{1,0} & \widehat{\alpha}_{1,1} & \widehat{\alpha}_{1,2} & 0 \\ \widehat{\alpha}_{2,0} & \widehat{\alpha}_{2,1} & \alpha_{2,2} & \alpha_{2,3} \\ \hline 0 & 0 & \alpha_{3,2} & \alpha_{3,3} \end{array}\right) \left(\begin{array}{c|ccc} 1 & 0 & 0 & 0 \\ \hline 0 & \gamma_{2,0} & -\sigma_{2,0} & 0 \\ 0 & \sigma_{2,0} & \gamma_{2,0} & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)$$

From: Gene H Golub <golub@stanford.edu>
Date: Sun, 19 Aug 2007 13:54:47 -0700 (PDT)
Subject: John Francis, Co-Inventor of QR

Dear Colleagues,

For many years, I have been interested in meeting J G F Francis, one of the co-inventors of the QR algorithm for computing eigenvalues of general matrices. Through a lead provided by the late Erin Brent and with the aid of Google, I finally made contact with him.

John Francis was born in 1934 in London and currently lives in Hove, near Brighton. His residence is about a quarter mile from the sea; he is a widower. In 1954, he worked at the National Research Development Corp (NRDC) and attended some lectures given by Christopher Strachey.
In 1955,'56 he was a student at Cambridge but did not complete a degree.
He then went back to NRDC as an assistant to Strachey where he got involved in flutter computations and this led to his work on QR.

After leaving NRDC in 1961, he worked at the Ferranti Corp and then at the University of Sussex. Subsequently, he had positions with various industrial organizations and consultancies. He is now retired. His interests were quite general and included Artificial Intelligence, computer languages, systems engineering. He has not returned to numerical computation.

He was surprised to learn there are many references to his work and that the QR method is considered one of the ten most important algorithms of the 20th century. He was unaware of such developments as TeX and Math Lab. Currently he is working on a degree at the Open University.

John Francis did remarkable work and we are all in his debt. Along with the conjugate gradient method, it provided us with one of the basic tools of numerical analysis.

Gene Golub

Figure 7: Posting by the late Gene Golub in NA Digest Sunday, August 19, 2007 Volume 07 : Issue 34. An article on the ten most important algorithms of the 20th century, published in SIAM News, can be found at http://www.uta.edu/faculty/rcli/TopTen/topten.pdf.

again preserves eigenvalues. Finally, from $\begin{pmatrix} \widehat{\alpha}_{2,1} \\ \widehat{\alpha}_{3,1} \end{pmatrix}$ one can compute $\gamma_{3,1}$ and $\sigma_{3,1}$ so that

$$\begin{pmatrix} \gamma_{3,1} & -\sigma_{3,1} \\ \sigma_{3,1} & \gamma_{3,1} \end{pmatrix}^T \begin{pmatrix} \widehat{\alpha}_{2,1} \\ \widehat{\alpha}_{3,1} \end{pmatrix} = \begin{pmatrix} \tilde{\alpha}_{2,1} \\ 0 \end{pmatrix}.$$

Then

$$\left(\begin{array}{cc|cc} \tilde{\alpha}_{0,0} & \tilde{\alpha}_{1,0} & 0 & 0 \\ \tilde{\alpha}_{1,0} & \tilde{\alpha}_{1,1} & \tilde{\alpha}_{2,1} & 0 \\ \hline 0 & \tilde{\alpha}_{2,1} & \tilde{\alpha}_{2,2} & \tilde{\alpha}_{2,3} \\ 0 & 0 & \tilde{\alpha}_{3,2} & \tilde{\alpha}_{3,3} \end{array}\right) = \left(\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 1 & 0 & \gamma_{3,2} & \sigma_{3,2} \\ 0 & 1 & -\sigma_{3,2} & \gamma_{3,2} \end{array}\right) \left(\begin{array}{cc|cc} \tilde{\alpha}_{0,0} & \tilde{\alpha}_{1,0} & 0 & 0 \\ \tilde{\alpha}_{1,0} & \tilde{\alpha}_{1,1} & \widehat{\tilde{\alpha}}_{2,1} & \widehat{\alpha}_{3,1} \\ \hline 0 & \widehat{\tilde{\alpha}}_{2,1} & \widehat{\alpha}_{2,2} & \widehat{\alpha}_{2,3} \\ 0 & \widehat{\alpha}_{3,1} & \widehat{\alpha}_{3,2} & \alpha_{3,3} \end{array}\right) \left(\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 1 & 0 & \gamma_{3,1} & -\sigma_{3,1} \\ 0 & 1 & \sigma_{3,1} & \gamma_{3,1} \end{array}\right)$$

The matrix Q is the orthogonal matrix that results from multiplying the different Givens' rotations together:

$$Q = \left(\begin{array}{cc|cc} \gamma_{1,0} & -\sigma_{1,0} & 0 & 0 \\ \sigma_{1,0} & \gamma_{1,0} & 0 & 0 \\ \hline 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right) \left(\begin{array}{c|cc|c} 1 & 0 & 0 & 0 \\ \hline 0 & \gamma_{2,0} & -\sigma_{2,0} & 0 \\ 0 & \sigma_{2,0} & \gamma_{2,0} & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \left(\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & \gamma_{3,1} & -\sigma_{3,1} \\ 0 & 0 & \sigma_{3,1} & \gamma_{3,1} \end{array}\right).$$

It is important to note that the first columns of Q is given by $\begin{pmatrix} \gamma_{1,0} \\ \sigma_{1,0} \\ 0 \\ 0 \end{pmatrix}$, which is exactly the same first column had Q been computed as in Section 5 (Equation 2). Thus, by the Implicit Q Theorem, the tridiagonal matrix that results from this approach is equal to the tridiagonal matrix that would be computed by applying the QR factorization from Section 5 with $A - \mu I$, $A - \mu I \rightarrow QR$ followed by the formation of $RQ + \mu I$ using the algorithm for computing RQ in Section 5.

The successive elimination of elements $\widehat{\alpha}_{i+1,i}$ is often referred to as *chasing the bulge* while the entire process that introduces the bulge and then chases it is known as a Francis Implicit QR Step. Obviously, the method generalizes to matrices of arbitrary size, as illustrated in Figure 8. An algorithm for the chasing of the bulge is given in Figure 9. (Note that in those figures T is used for A, something that needs to be made consistent in these notes, eventually.) In practice, the tridiagonal matrix is not stored as a matrix. Instead, its diagonal and subdiagonal are stored as vectors.

6.4 A complete algorithm

This last section shows how one iteration of the QR algorithm can be performed on a tridiagonal matrix by implicitly shifting and then "chasing the bulge". All that is left to complete the algorithm is to note that

- The shift μ_k can be chosen to equal $\alpha_{n-1,n-1}$ (the last element on the diagonal, which tends to converge to the eigenvalue smallest in magnitude). In practice, choosing the shift to be an eigenvalue of the bottom-right 2×2 matrix works better. This is known as the *Wilkinson Shift*.
- If an element of the subdiagonal (and corresponding element on the superdiagonal) becomes small enough, it can be considered to be zero and the problem deflates (decouples) into two smaller tridiagonal matrices. Small is often taken to means that $|\alpha_{i+1,i}| \leq \epsilon(|\alpha_{i,i}| + |\alpha_{i+1,i+1}|)$ where ϵ is some quantity close to the machine epsilon (unit roundoff).
- If $A = QTQ^T$ reduced A to the tridiagonal matrix T before the QR algorithm commensed, then the Givens' rotations encountered as part of the implicitly shifted QR algorithm can be applied from the

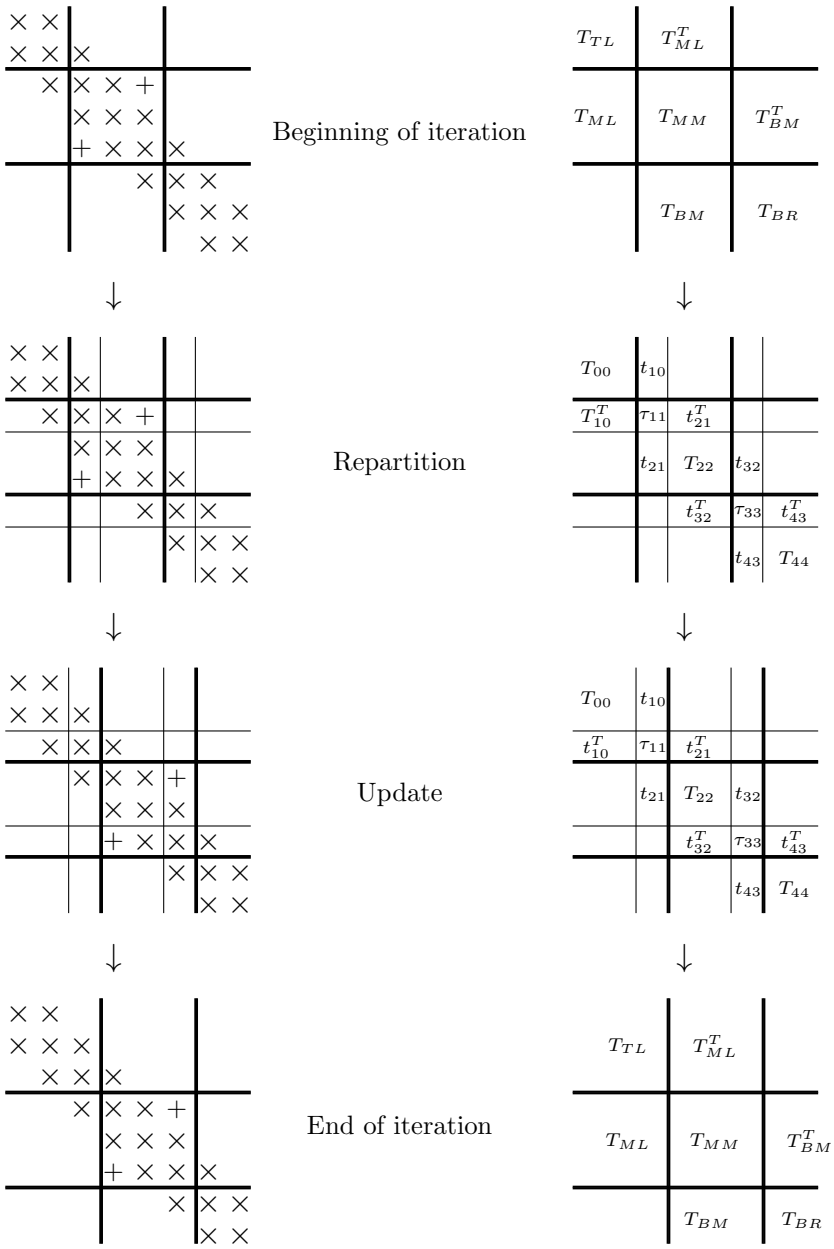

Figure 8: One step of "chasing the bulge" in the implicitly shifted symmetric QR algorithm.

right to the appropriate columns of Q so that upon completion Q is overwritten with the eigenvectors of A. Notice that applying a Givens' rotation to a pair of columns of Q requires $O(n)$ computation per Givens rotation. For each Francis implicit QR step $O(n)$ Givens' rotations are computed, making the application of Givens' rotations to Q of cost $O^(n^2)$ per iteration of the implicitly shifted QR algorithm. Typically a few (2-3) iterations are needed per eigenvalue that is uncovered (by deflation) meaning that $O(n)$ iterations are needed. Thus, the QR algorithm is roughly of cost $O(n^3)$ if the eigenvalues are accumulated (in addition to the cost of forming the Q from the reduction to tridiagonal form, which takes another $O(n^3)$ operations.)

Algorithm: $T := \text{CHASEBULGE}(T)$

Partition $T \rightarrow \left(\begin{array}{c|c|c} T_{TL} & \star & \star \\ \hline T_{ML} & T_{MM} & \star \\ \hline 0 & T_{BM} & T_{BR} \end{array}\right)$

where T_{TL} is 0×0 and T_{MM} is 3×3

while $m(T_{BR}) > 0$ **do**

Repartition

$$\left(\begin{array}{c|c|c} T_{TL} & \star & 0 \\ \hline T_{ML} & T_{MM} & \star \\ \hline 0 & T_{BM} & T_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c|c|c} T_{00} & \star & 0 & 0 & 0 \\ \hline t_{10}^T & \tau_{11} & \star & 0 & 0 \\ \hline 0 & t_{21} & T_{22} & \star & 0 \\ \hline 0 & 0 & t_{32}^T & \tau_{33} & \star \\ \hline 0 & 0 & 0 & t_{43} & T_{44} \end{array}\right)$$

where τ_{11} and τ_{33} are scalars
(during final step, τ_{33} is 0×0)

Compute (γ, σ) s.t. $G_{\gamma,\sigma}^T t_{21} = \begin{pmatrix} \tau_{21} \\ 0 \end{pmatrix}$, and **assign** $t_{21} = \begin{pmatrix} \tau_{21} \\ 0 \end{pmatrix}$

$T_{22} = G_{\gamma,\sigma}^T T_{22} G_{\gamma,\sigma}$

$t_{32}^T = t_{32}^T G_{\gamma,\sigma}$ (not performed during final step)

Continue with

$$\left(\begin{array}{c|c|c} T_{TL} & \star & 0 \\ \hline T_{ML} & T_{MM} & \star \\ \hline 0 & T_{BM} & T_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c|c|c} T_{00} & \star & 0 & 0 & 0 \\ \hline t_{10}^T & \tau_{11} & \star & 0 & 0 \\ \hline 0 & t_{21} & T_{22} & \star & 0 \\ \hline 0 & 0 & t_{32}^T & \tau_{33} & \star \\ \hline 0 & 0 & 0 & t_{43} & T_{44} \end{array}\right)$$

endwhile

Figure 9: Chasing the bulge.

- If an element on the subdiagonal becomes zero (or very small), and hence the corresponding element of the superdiagonal, then the problem can be *deflated*: If

$$T = \left(\begin{array}{c|c} T_{00} & 0 \\ \hline 0 & T_{11} \end{array}\right)$$

$$\left(\begin{array}{ccccc|ccc} \times & \times & & & & & & \\ \times & \times & \times & & & & & \\ & \times & \times & \times & & & & \\ & & \times & \times & \times & & & \\ & & & \times & \times & 0 & & \\ \hline & & & & 0 & \times & \times & \\ & & & & & \times & \times & \times \\ & & & & & & \times & \times \end{array}\right)$$

then

  - The computation can continue separately with T_{00} and T_{11}.
  - One can pick the shift from the bottom-right of T_{00} as one continues finding the eigenvalues of T_{00}, thus accelerating the computation.
  - One can pick the shift from the bottom-right of T_{11} as one continues finding the eigenvalues of T_{11}, thus accelerating the computation.

– One must continue to accumulate the eigenvectors by applying the rotations to the appropriate columns of Q.

Because of the connection between the QR algorithm and the Inverse Power Method, subdiagonal entries near the bottom-right of T are more likely to converge to a zero, so most deflation will happen there.

7 Further Reading

7.1 More on reduction to tridiagonal form

The reduction to tridiagonal form can only be partially cast in terms of matrix-matrix multiplication [5]. This is a severe hindrance to high performance for that first step towards computing all eigenvalues and eigenvector of a symmetric matrix. Worse, a considerable fraction of the total cost of the computation is in that first step.

For a detailed discussion on the blocked algorithm that uses FLAME notation, we recommend [8]

Field G. Van Zee, Robert A. van de Geijn, Gregorio Quintana-Ort, G. Joseph Elizondo.
Families of Algorithms for Reducing a Matrix to Condensed Form.
ACM Transactions on Mathematical Software (TOMS) , Vol. 39, No. 1, 2012

(Reduction to tridiagonal form is one case of what is more generally referred to as "condensed form".)

7.2 Optimizing the tridiagonal QR algorithm

As the Givens' rotations are applied to the tridiagonal matrix, they are also applied to a matrix in which eigenvectors are accumulated. While one Implicit Francis Step requires $O(n)$ computation, this accumulation of the eigenvectors requires $O(n^2)$ computation with $O(n^2)$ data. We have learned before that this means the cost of accessing data dominates on current architectures.

In a recent paper, we showed how accumulating the Givens' rotations for several Francis Steps allows one to attain performance similar to that attained by a matrix-matrix multiplication. Details can be found in [7]:

Field G. Van Zee, Robert A. van de Geijn, Gregorio Quintana-Ortí.
Restructuring the Tridiagonal and Bidiagonal QR Algorithms for Performance.
ACM Transactions on Mathematical Software (TOMS, Vol. 40, No. 3, 2014.

8 Other Algorithms

8.1 Jacobi's method for the symmetric eigenvalue problem

(Not to be mistaken for the Jacobi iteration for solving linear systems.)

The oldest algorithm for computing the eigenvalues and eigenvectors of a matrix is due to Jacobi and dates back to 1846 [6]. This is a method that keeps resurfacing, since it parallelizes easily.

The idea is as follows: Given a symmetric 2×2 matrix

$$A_{31} = \begin{pmatrix} \alpha_{11} & \alpha_{31} \\ \alpha_{31} & \alpha_{j,j} \end{pmatrix}$$

There exists a rotation (which is of course unitary)

$$J_{31} = \begin{pmatrix} \gamma_{11} & -\sigma_{31} \\ \sigma_{31} & \alpha_{33} \end{pmatrix}$$

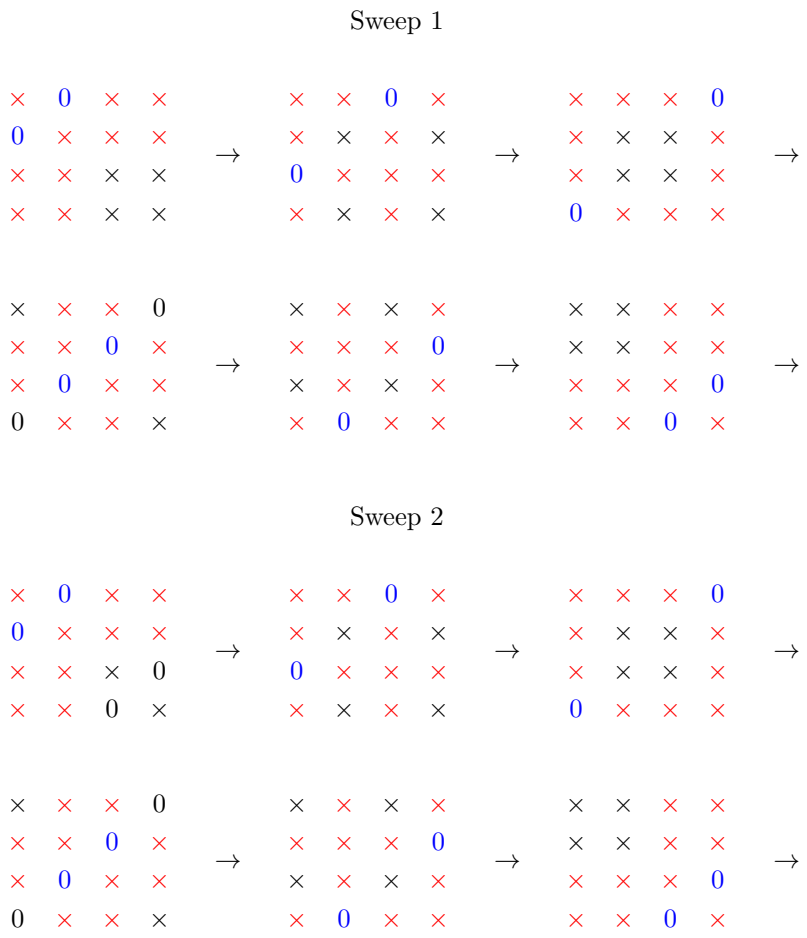

Figure 10: Column-cyclic Jacobi algorithm.

such that

$$J_{31}A_{31}J_{31}^T = \begin{pmatrix} \gamma_{11} & -\sigma_{31} \\ \sigma_{31} & \gamma_{33} \end{pmatrix} \begin{pmatrix} \alpha_{11} & \alpha_{31} \\ \alpha_{31} & \alpha_{33} \end{pmatrix} \begin{pmatrix} \gamma_{11} & -\sigma_{31} \\ \sigma_{31} & \gamma_{33} \end{pmatrix}^T = \begin{pmatrix} \widehat{\alpha}_{11} & 0 \\ 0 & \widehat{\alpha}_{33} \end{pmatrix}.$$

We know this exists since the Spectral Decomposition of the 2×2 matrix exists. Such a rotation is called a Jacobi rotation. (Notice that it is different from a Givens' rotation because it diagonalizes a 2×2 matrix when used as a unitary similarity transformation. By contrast, a Givens' rotation zeroes an element when applied from one side of a matrix.)

Exercise 10. *In the above discussion, show that* $\alpha_{11}^2 + 2\alpha_{31}^2 + \alpha_{31}^2 = \widehat{\alpha}_{11}^2 + \widehat{\alpha}_{33}^2$.

Jacobi rotation rotations can be used to selectively zero off-diagonal elements by observing the following:

$$JAJ^T = \left(\begin{array}{c|c|c|c|c} I & 0 & 0 & 0 & 0 \\ \hline 0 & \gamma_{11} & 0 & -\sigma_{31} & 0 \\ \hline 0 & 0 & I & 0 & 0 \\ \hline 0 & \sigma_{31} & 0 & \gamma_{33} & 0 \\ \hline 0 & 0 & 0 & 0 & I \end{array}\right) \left(\begin{array}{c|c|c|c|c} A_{00} & a_{10} & A_{20}^T & a_{30} & A_{40}^T \\ \hline a_{10}^T & \alpha_{11} & a_{21}^T & \alpha_{31} & a_{41}^T \\ \hline A_{20} & a_{21} & A_{22} & a_{32} & A_{42}^T \\ \hline a_{30}^T & \alpha_{31} & a_{32}^T & \alpha_{33} & a_{43}^T \\ \hline A_{40} & a_{41} & A_{42} & a_{43} & A_{44} \end{array}\right) \left(\begin{array}{c|c|c|c|c} I & 0 & 0 & 0 & 0 \\ \hline 0 & \gamma_{11} & 0 & -\sigma_{13} & 0 \\ \hline 0 & 0 & I & 0 & 0 \\ \hline 0 & \sigma_{31} & 0 & \gamma_{33} & 0 \\ \hline 0 & 0 & 0 & 0 & I \end{array}\right)^T$$

$$= \left(\begin{array}{c|c|c|c|c} A_{00} & \widehat{a}_{10} & A_{20}^T & \widehat{a}_{30} & A_{40}^T \\ \hline \widehat{a}_{10}^T & \widehat{\alpha}_{11} & \widehat{a}_{21}^T & 0 & \widehat{a}_{41}^T \\ \hline A_{20} & \widehat{a}_{21} & A_{22} & \widehat{a}_{32} & A_{42}^T \\ \hline \widehat{a}_{30}^T & 0 & \widehat{a}_{32}^T & \widehat{\alpha}_{33} & \widehat{a}_{43}^T \\ \hline A_{40} & \widehat{a}_{41} & A_{42} & \widehat{a}_{43} & A_{44} \end{array}\right) = \widehat{A},$$

where

$$\left(\begin{array}{cc} \gamma_{11} & -\sigma_{31} \\ \sigma_{31} & \gamma_{33} \end{array}\right) \left(\begin{array}{c|c|c} a_{10}^T & a_{21}^T & a_{41}^T \\ a_{30}^T & a_{32}^T & a_{43}^T \end{array}\right) = \left(\begin{array}{c|c|c} \widehat{a}_{10}^T & \widehat{a}_{21}^T & \widehat{a}_{41}^T \\ \widehat{a}_{30}^T & \widehat{a}_{32}^T & \widehat{a}_{43}^T \end{array}\right).$$

Importantly,

$$\begin{array}{rcl} a_{10}^T a_{10} + a_{30}^T a_{30} & = & \widehat{a}_{10}^T \widehat{a}_{10} + \widehat{a}_{30}^T \widehat{a}_{30} \\ a_{21}^T a_{21} + a_{32}^T a_{32} & = & \widehat{a}_{21}^T \widehat{a}_{21} + \widehat{a}_{32}^T \widehat{a}_{32} \\ a_{41}^T a_{41} + a_{43}^T a_{43} & = & \widehat{a}_{41}^T \widehat{a}_{41} + \widehat{a}_{43}^T \widehat{a}_{43}. \end{array}$$

What this means is that if one defines $\text{off}(A)$ as the square of the Frobenius norm of the off-diagonal elements of A,

$$\text{off}(A) = \|A\|_F^2 - \|\text{diag}(A)\|_F^2,$$

then $\text{off}(\widehat{A}) = \text{off}(A) - 2\alpha_{31}^2$.

- The good news: every time a Jacobi rotation is used to zero an off-diagonal element, $\text{off}(A)$ decreases by twice the square of that element.
- The bad news: a previously introduced zero may become nonzero in the process.

The original algorithm developed by Jacobi searched for the largest (in absolute value) off-diagonal element and zeroed it, repeating this processess until all off-diagonal elements were small. The algorithm was applied by hand by one of his students, Seidel (of Gauss-Seidel fame). The problem with this is that searching for the largest off-diagonal element requires $O(n^2)$ comparisons. Computing and applying one Jacobi rotation as a similarity transformation requires $O(n)$ flops. Thus, for large n this is not practical. Instead, it can be shown that zeroing the off-diagonal elements by columns (or rows) also converges to a diagonal matrix. This is known as the column-cyclic Jacobi algorithm. We illustrate this in Figure 10.

8.2 The Method of Multiple Relatively Robust Representations (MRRR)

Even once the problem has been reduced to tridiagonal form, the computation of the eigenvalues and eigenvectors via the QR algorithm requires $O(n^3)$ computations. A method that reduces this to $O(n^2)$ time (which can be argued to achieve the lower bound for computation, within a constant, because the n vectors must be at least written) is achieved by the Method of Multiple Relatively Robust Representations (MRRR) by Dhillon and Partlett [2, 1, 3, 4]. The details of that method go beyond the scope of this note.

References

[1] I. S. Dhillon. *A New $O(n^2)$ Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem.* PhD thesis, Computer Science Division, University of California, Berkeley, California, May 1997. Available as UC Berkeley Technical Report No. UCB//CSD-97-971.

[2] I. S. Dhillon. Reliable computation of the condition number of a tridiagonal matrix in $O(n)$ time. *SIAM J. Matrix Anal. Appl.*, 19(3):776–796, July 1998.

[3] I. S. Dhillon and B. N. Parlett. Multiple representations to compute orthogonal eigenvectors of symmetric tridiagonal matrices. *Lin. Alg. Appl.*, 387:1–28, August 2004.

[4] Inderjit S. Dhillon, Beresford N. Parlett, and Christof Vömel. The design and implementation of the MRRR algorithm. *ACM Transactions on Mathematical Software*, 32(4):533–560, December 2006.

[5] Jack J. Dongarra, Sven J. Hammarling, and Danny C. Sorensen. Block reduction of matrices to condensed forms for eigenvalue computations. *Journal of Computational and Applied Mathematics*, 27, 1989.

[6] C. G. J. Jacobi. Über ein leichtes Verfahren, die in der Theorie der Säkular-störungen vorkommenden Gleichungen numerisch aufzulösen. *Crelle's Journal*, 30:51–94, 1846.

[7] Field G. Van Zee, Robert A. van de Geijn, and Gregorio Quintana-Ortí. Restructuring the tridiagonal and bidiagonal qr algorithms for performance. *ACM Transactions on Mathematical Software*, 40(3):18:1–18:34, April 2014.

[8] Field G. Van Zee, Robert A. van de Geijn, Gregorio Quintana-Ortí, and G. Joseph Elizondo. Families of algorithms for reducing a matrix to condensed form. *ACM Trans. Math. Soft.*, 39(1), 2012.