Orthogonal Iteration

• How can we use the power method to compute more than one eigenvalue?

• How about

1. Pick a random $n \times r$ matrix $Y_0$.

2. For $k = 0, 1, 2, \ldots$:

$$Z_{k+1} = AY_k$$

$$Y_{k+1} = \frac{Z_{k+1}}{\|Z_{k+1}\|}$$

• This is no good. The columns of $Y_k$ do not interact with each other, and so we run $r$ copies of the scalar power method. Each column of $Y_k$ will converge to the same dominant eigenvector.

• We need to keep those columns linearly independent.

• How independent?

• How about orthogonal?
• This gives rise to **Orthogonal Iteration**:

1. Suppose $Q_0 = Q$ is $n \times r$ with orthonormal columns, i.e.,
   \[
   Q^T Q = I_r
   \]

2. For $k = 0, 1, 2, \ldots$:
   
   compute $Z_{k+1} = AQ_k$
   
   factor $Z_{k+1} = Q_{k+1} R_{k+1}$

   where
   
   $R_k$ is upper triangular and $Q_k^T Q_k = I_r$.

• Note that as far as the first $s$ columns of $Q_k$ are concerned, this is just orthogonal iteration with $s$ columns.

• We have $r$ nested orthogonal iterations.

\[\mathcal{Z}\]  How about $r = n$?

• We would find all eigenvectors.
• It turns out that Orthogonal Iteration with \( r = n \) is equivalent to running the \textbf{QR iteration}:

1. \( T_0 = A \)

2. For \( k = 0, 1, 2, \ldots \):
   
   Factor \( T_k = Q_k R_k \)
   
   Compute \( T_{k+1} = R_k Q_k \)

   or, more succinctly

2. For \( k = 0, 1, 2, \ldots \):
   
   Compute \( T = QR \)
   
   Overwrite \( T = RQ \)

• Or yet more succinctly, if you don’t mind overwriting \( A \):

2. For \( k = 0, 1, 2, \ldots \):
   
   Compute \( A = QR \)
   
   Overwrite \( A = RQ \)

• matlab demo

• Why does this work?

• go back to orthogonal iteration:
\[ Q_0 = I \]

For \( k = 1, 2, 3, \ldots \)

\[ Z_k = R_k Q_{k-1} \]
\[ Z_k = Q_k R_k \]

- Let

\[ T_k = Q_k^T A Q_k. \]

\( T_k \) is similar to \( A \)!

- Also note that

\[ AQ_{k-1} = Z_k = Q_k R_k \]

- \( T_k \) is obtained from \( T_{k-1} \) by the QR iteration. To see this note that the QR factorization of \( T_{k-1} \) is given by

\[ T_{k-1} = Q_{k-1}^T A Q_{k-1} = Q_{k-1}^T Z_k = Q_{k-1}^T Q_k R_k = Q R \]

where

\[ Q = Q_{k-1}^T Q_k \quad \text{and} \quad R = R_k. \quad (1) \]
Now consider $T_k$:

$$T_k = A_k^T A Q_k$$

$$= A_k^T A Q_{k-1} Q_{k-1}^T Q_k$$

$$= Q_k^T Z_k Q_{k-1}^T Q_k$$

$$= Q_k^T Q_{k-1} R_k Q_{k-1}^T Q_k$$

$$= R_k Q_{k-1}^T Q_k$$

$$= R Q$$

where $Q$ and $R$ are given in (1).

- Note that every step of this algorithm requires $O(n^3)$ operations.
- The QR algorithm starts with this idea and greatly refines it.
QR Algorithm Overview

• The QR Algorithm computes the eigenvalues (and the eigenvectors, if required) of a general full matrix.

• Variants of the QR algorithm exist for symmetric or sparse matrices.

• The QR algorithm is much more complicated than its counterpart, Gaussian Elimination, the basic technique for solving linear systems.


• However, as a first introduction you may want to read the classic article by David S. Watkins, Understanding the QR Algorithm, SIAM Review, 1982, Vol. 24, No. 4: pp. 427-440.

• We sometimes cover the QR algorithm in depth in Math 6610. If you contemplate taking that course check with the instructor before the semester starts.

• Studying the QR algorithm is well worth your time and effort since it embodies most, if not all, of the key concepts of numerical linear algebra.
Outline of the QR Algorithm

- The basic idea is to apply orthogonal similarity transforms to convert $A$ to **Real Schur Form**:

$$R = Q^T AQ = \begin{bmatrix}
R_{11} & R_{12} & \cdots & R_{1m} \\
0 & R_{22} & \cdots & R_{2m} \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & R_{mm}
\end{bmatrix}$$

where $R$ is real and upper block triangular, $Q$ is orthogonal, and each $R_{ii}$ is either a $1 \times 1$ matrix or a $2 \times 2$ matrix having conjugate complex eigenvalues.

- A block triangular matrix whose diagonal blocks are $1 \times 1$ or $2 \times 2$ is called **upper quasi-triangular**.

- The eigenvalues of the $R_{ii}$ are the eigenvalues of $A$.

- The Real Schur form exists for every square matrix $A$.

- The first step of the $QR$ Algorithm consists of finding an orthogonal similarity transform that takes $A$ to **upper Hessenberg Form**.

- A matrix $A$ is upper Hessenberg if

$$i > j + 1 \quad \Rightarrow \quad a_{ij} = 0.$$
also be non-zero. For example, a $6 \times 6$ upper Hessenberg matrix has the form

$$H = \begin{bmatrix}
x & x & x & x & x & x \\
x & x & x & x & x & x \\
0 & x & x & x & x & x \\
0 & 0 & x & x & x & x \\
0 & 0 & 0 & x & x & x \\
0 & 0 & 0 & 0 & x & x
d\end{bmatrix}$$

- The initial reduction to upper Hessenberg form can be computed in a finite number of operations. The algorithm is based on using Householder reflection based orthogonal similarity transforms to go from column to column, similarly to the computation of the $QR$ factorization.

- **Exercise:** work out the details of this initial stage.

- An upper Hessenberg matrix is **unreduced** if all subdiagonal entries are non-zero.

- The second stage of the QR Algorithm is an iteration that consists of applying orthogonal transformations, based on Givens Rotations, that reduce the upper Hessenberg matrix to Real Schur Form.

- The iterations in the second stage involve a double shift of origin which accelerates the reduction of the upper Hessenberg matrix to quasi-triangular form.
Ingredients and Principles

• Even though eigenvalues and eigenvectors may be complex, the arithmetic is real throughout.

• All similarity transforms are orthogonal, and those in stage 2 preserve the upper Hessenberg structure.

• The initial stage requires $O(n^3)$ operations.

• Every step of the iteration in the second stage requires only $O(n^2)$ operations.

• Accomplishing each iteration in $O(n^2)$ operations is the most complicated part of the algorithm. The breakthrough that made this possible was a technique developed by John Francis and published in 1961. This is the Francis QR step, Algorithm 7.5.1 on page 390 in Golub/van Loan.

• The following outline is taken (and modified) from Algorithm 7.5.2 of Golub/van Loan.
The QR Algorithm

Let \( \text{tol} \) be a tolerance greater than the roundoff unit.

Compute the Hessenberg Reduction

\[
H = U_0^T A U_0
\]

where \( H \) is upper Hessenberg and \( U_0 \) is orthogonal.

Set \( q = 0 \)

until \( q = n \)

Set to zero all subdiagonal entries of \( H \) that satisfy

\[
|h_{i,i-1}| \leq \text{tol}(|h_{ii}| + |h_{i-1,i-1}|)
\]

Find the largest nonnegative \( q \) and the smallest non-negative \( p \) such that

\[
H = \begin{pmatrix}
& p & n - p - q & q \\
p & H_{11} & H_{12} & H_{13} \\
q & 0 & H_{22} & H_{23} \\
& 0 & 0 & H_{33}
\end{pmatrix}
\]

where \( H_{33} \) is upper quasi-triangular and \( H_{22} \) is unreduced.

If \( q < n \) perform a Francis QR step on \( H_{22} \)

Upper triangularize all \( 2 \times 2 \) blocks in \( H \) that have real eigenvalues.
• According to Golub/van Loan, The algorithm requires $25n^3$ flops if the eigenvectors are computed and $10n^3$ flops if only the eigenvalues are computed. These counts are very approximate and based on the empirical observation that on average only two Francis iterations are required before lower right $1 \times 1$ or $2 \times 2$ submatrix of $H_{22}$ decouples.

• The QR Algorithm is extremely sophisticated, but it grew out of simple ideas in natural steps. Golub/van Loan and Watkins both explain this very well.
Summary

- These are some of the key ingredients of the design and analysis of the QR algorithm:
  - Accomplish each task by multiplying with an orthogonal matrix.
  - Work on subproblems and embed the required matrices in the identity matrix.
  - power iterations
  - orthogonal iterations
  - many nested orthogonal iterations
  - shift of origin
  - real Schur form
  - Use of Householder reflections and Givens rotations
  - $\mathcal{O}(n^2)$ effort per step
  - real arithmetic
  - start with upper Hessenberg
  - keep it upper Hessenberg
  - Implicit Q Theorem (underlying Francis step).
• The story has a happy ending. In 2009 the following item was posted by Frank Uhlig on the Numerical Analysis Bulletin board:

From: Frank Uhlig <uhligfd@auburn.edu>  
Date: Wed, 25 Mar 2009 08:48:14 -0500  
Subject: John Francis of QR

**John Francis and 50 years of QR**

John Francis submitted his first QR paper almost 50 years ago in October 1959. By 1962 he had left the NA field. When his algorithm was judged one of the top ten algorithms of the 20th century in 2000 by Jack Dongarra and Francis Sullivan, nobody alive in the mathematics community had ever seen John Francis or knew where or if he lived. Gene Golub and Frank Uhlig independently tracked John Francis down, joined forces, and visited and interviewed him over the last couple of years.

When first contacted, John Francis had no idea about QR’s impact. He is 74 years old now and well. Re QR, he remembers his math and computational work of 50 years ago clearly. John Francis will be the lead-off speaker at a mini symposium, held in his honor, at the 23rd Biennial Conference on Numerical Analysis, June 23rd - 26th 2009 in Glasgow to which everyone is cordially invited.