EIGENVALUE ALGORITHMS

AXEL RUHE\(^1\)

April 2, 2007

\(^1\)Department of Numerical Analysis and Computer Science, Royal Institute of Technology, SE-10044 Stockholm, Sweden (ruhe@kth.se). Material given to graduate course in Numerical Linear Algebra.
Preface

This is some of the material I find worth covering in a course for graduate students in Scientific Computation who will need to understand algorithms based on linear algebra and use them in different applications.

For the theory, I like the exposition by Stewart and Sun [5], but you can find most of the material in the textbook by Demmel [2]. Algorithms, as the state of the art was in year 2000, are described in the Templates for Eigenvalue book [1]. A more detailed description of the algorithms involved is given in the book series of Stewart, see [4]! We also need to follow some of the later developments, global algorithms based on matrix sign functions, fast algorithms for tridiagonals and high relative precision algorithms to name three interesting topics.

Stockholm April 2, 2007

Axel Ruhe
Chapter 1

Eigenvalues

In many scientific and engineering contexts one needs to solve an algebraic eigenvalue problem,
\[(A - \lambda B)x = 0, \quad (1.1)\]
for eigenvalues \(\lambda_k\) and eigenvectors \(x_k\). Its solution is used as the first step on the way from a static to a dynamic description of a system of interacting entities. Think of a network of electric components or a mechanical construction of masses and springs! Small perturbations of a static, equilibrium, position will be formed as eigenvectors and move as the eigenvalues indicate.

The foremost mathematical modeling tool is the Laplace transform, where we map a description in time and space to one involving decay rates and oscillation frequencies, these are the real and imaginary parts of the eigenvalues. In this realm we use eigenvalues to get the solution of a linear system of ordinary differential equations,
\[\dot{x} = Ax, \quad x(0) = a,\]
over time as a linear combination of eigensolutions,
\[x(t) = \Sigma \alpha_k x_k e^{\lambda_k t}.\]

1.1 Mathematical theory

1.1.1 The Schur decomposition

We can transform any square matrix to triangular form by means of an unitary (or orthogonal) similarity. This is the consequence of the Schur theorem,

**Theorem 1 (Schur theorem)** Every square matrix \(A\) can be transformed
\[A = UTU^H\]
where \(U\) is unitary and \(T\) is upper triangular with eigenvalues in the diagonal in any chosen order.
Proof: The proof is constructive. Find an eigenvalue eigenvector pair, transform the first row and column to the form wanted, and repeat the procedure on a matrix of smaller size.

Take \( \lambda_1 \) an eigenvalue and \( x \) an eigenvector of unit norm, \( \|x\|_2 = 1 \). Add \( n-1 \) column vectors \( X_{n-1} \) to make a unitary matrix, \( U_1 = [x, X_{n-1}] \).

Now

\[
Ax = x\lambda_1 \\
AU_1 = [x\lambda_1, AX_{n-1}] \\
U_1^HAU_1 = \begin{bmatrix} \lambda_1 & A'_{1,n-1} \\ 0 & A'_{n-1,n-1} \end{bmatrix}
\]

The zeros in the lower left are there because \( U \) is unitary, its first column \( x \) is orthogonal to the remaining columns in \( X_{n-1} \).

We have now made the first step, repeat the same procedure on the lower right block \( A'_{n-1,n-1} \) giving \( \lambda_2 \) and \( U_2 \), and continue down to the right, until we have an upper triangular matrix \( T \) and a unitary \( U = U_1U_2\ldots U_{n-1} \), with parts of the unit matrix added to each \( U_k \), to make it \( n \times n \). This ends the proof.

1.2 Transformation algorithms

For matrices of moderate size, essentially those that can be stored as a full array in the computer, the standard way to compute eigenvalues is to apply similarity transformations. Most often we insist on doing unitary (orthogonal in the real case) similarity transformations, since they do not change the eigenvalues. They are also backwards stable, making the computed solution the exact solution of a problem with moderately perturbed data. We will later see more general transformations, when discussing spectral transformations and preconditioning.

The aim of a transformation algorithm is to bring the matrix into a form from which the solution can be read out. The first part can be done in a systematic finite algorithm, that makes certain elements zero, then follows an iterative part for the final approach.

We can make statements about the numerical behavior of an algorithm, by showing that it is equivalent to another with known behavior. Here we can use the theory of Krylov sequences, to explain a transformation algorithm seen as an iteration. We use the theory of polynomials to explain the convergence of Krylov type algorithms.

Theorem 2 (Hessenberg uniqueness) Any decomposition

\[
AQ = QH
\]

where \( A \) is a given \( n \times n \) matrix, \( Q \) is an \( n \times k \) matrix with orthonormal columns, \( Q^HQ = I_k \), and \( H \) is a \( k \times k \) unreduced Hessenberg matrix, with real positive subdiagonal \( h_{j+1,j} > 0, j = 1, \ldots, k-1 \), is uniquely defined by the first column \( q_1 \) of the basis \( Q \).
The theorem is proved by the Arnoldi algorithm started at the given first column \( q_1 \). If it runs its full course, \( k = n \), the matrix \( A \) is non-derogatory, having only one eigenvector for each eigenvalue and \( q_1 \) has components from all eigenvectors. Otherwise the minimal polynomial of \( A \) at \( q_1 \) has degree \( k \).

We use the Hessenberg uniqueness theorem to explain the Householder reduction to Hessenberg form and the implicitly shifted QR algorithm. It is also the basis of implicitly restarted Arnoldi and rational Krylov algorithms.

An important consequence of the Hessenberg uniqueness theorem is that we can use the theory of Krylov sequences to explain the behavior of any algorithm involving transformation to Hessenberg form, also those that do not explicitly let the matrix operate on a sequence of vectors.

We can also say that any decomposition into a matrix of lower dimension is a kind of Krylov sequence algorithm.

**Theorem 3 (Krylov equivalence)** Any decomposition

\[
AX_k = X_{k+b} K_{k+b,k}
\]

where \( A \) is a given \( n \times n \) matrix, \( X \) is a matrix with linearly independent columns, and \( K \) is a \((k+b) \times k\) matrix, is equivalent to a block Krylov algorithm started on a \( b \) dimensional subspace of the span of \( X_k \).

**Proof:** We can as well assume that the basis is unitary, just do a QR factorization \( X = QR \) and replace \( K \) by \( R_{k+b,k+b} K R^{-1}_{k+b,k} \). Apply a sequence of unitary reflection similarities to \( K \). The first reflection affects the first \( k \) coordinates and zeroes out the first \( k - 1 \) elements of the last row \( k + b \). The next leaves the last column unaffected and zeroes out the leading \( k - 2 \) elements of the second last row \( k + b - 1 \). Continue until \( k \) reflections are applied. Note that subsequent reflections do not affect the zeros already introduced and that the final \( K \) matrix has an upper triangular block at the bottom, just like when a block Hessenberg reduction is performed. For details see [3]!

### 1.2.1 Initial reduction: Householder algorithm

### 1.2.2 Final iteration: QR algorithm

The first phase of the transformation algorithm was finite, and needed \( n - 2 \) steps to get the matrix into Hessenberg form. Now in the second phase, the Hessenberg form will be retained, while we will make the subdiagonal elements smaller and smaller. The standard algorithm is the QR algorithm, which does the similarities by doing a sequence of QR factorizations of shifted matrices.

The basic QR algorithm is

**Algorithm QR, unshifted**

Start \( A_1 = A \) (Hessenberg matrix), \( U_1 = I \) (transformation).
For \( k = 1, \ldots, \) do
1. Factorize $A_k = Q_k R_k$ giving $Q_k$, orthogonal, and $R_k$, upper triangular.

2. Multiply $A_{k+1} = R_k Q_k$

3. Accumulate $U_{k+1} = U_k Q_k$

END.

We see that

$$A_{k+1} = R_k Q_k = Q_k^H A_k Q_k$$  \hfill (1.4)

an orthogonal similarity. The accumulation makes the final

$$A_{k+1} = U_k^H A U_k$$

In order to study the convergence, we will relate the QR algorithm to the power method and inverse iteration. We do this by looking at the accumulated unitary transformation $U_k$ and note that it is the Q factor of a power of the original matrix. Multiply the accumulated transformation with the product $S_k$ of the upper triangular matrices in reverse order,

$$U_k S_k = Q_1 Q_2 \ldots Q_k R_k R_{k-1} \ldots R_1$$

$$= Q_1 \ldots Q_{k-1} A_k R_{k-1} \ldots R_1$$

$$= Q_1 \ldots Q_{k-1} A_k Q_k^H R_{k-1} \ldots R_1$$

$$= Q_1 \ldots Q_{k-2} A_{k-1} Q_{k-1} R_{k-1} \ldots R_1$$

$$= A_1 Q_1 \ldots Q_{k-2} Q_{k-1} R_{k-1} \ldots R_1$$

$$= A_1^2 Q_1 \ldots Q_{k-2} R_{k-2} \ldots R_1$$

$$= \ldots$$

From this factorization we can see how the columns of the accumulated transformation matrix $U_k$ approach those of the Schur decomposition. Look at the first column,

$$A^k e_1 = U_k S_k e_1 = (U_k^k)_{1,1} s_{1,1}$$

the result of $k$ iterations of the power method started at the basis vector $e_1$.

Likewise the last column is related to the inverse power method since,

$$S_k^{-1} U_k^{-1} = A^{-k}$$

$$U_k S_k^{-H} = (A^H)^{-k}$$

$$U_k S_k^{-H} = (A^H)^{-k} e_n$$

the result of $k$ iterations of inverse iteration with $A^H$ started at the last basis vector $e_n$.

It is slightly more complicated to show that the whole transformation converges to the unitary matrix of Schur vectors. This assumes that the eigenvalues are ordered after strictly decreasing absolute values, see [4] p. 78 Th 2.2!

Let us formulate a simple variant of the shifted algorithm.
Algorithm QR, with explicit shifts

Start $A_1$ (Hessenberg matrix), $U_1 = I$ (transformation).
For $k = 1, \ldots$, do

1. Choose shift $\sigma_k$
2. Factorize $A_k - \sigma_k I = Q_k R_k$ (Shifted matrix)
3. Multiply $A_{k+1} = R_k Q_k + \sigma_k I$ (Restore shift)
4. Accumulate $U_{k+1} = U_k Q_k$

End.

The new $A_{k+1}$ is still an orthogonal similarity of $A_k$

\[ A_{k+1} = R_k Q_k + \sigma_k I = (Q_k^T (A_k - \sigma_k I)) Q_k + \sigma_k I = Q_k^T A_k Q_k \]

Now the accumulated transformation matrix $U_k$ is the Q factor of a the $k$ degree polynomial $s_k(A)$ where

\[ s_k(\lambda) = (\lambda - \sigma_1)(\lambda - \sigma_2) \ldots (\lambda - \sigma_k) \]

has its roots at the shifts $\sigma_i$.

Implicit shifts If the first $A_1$ is of Hessenberg form, this form is retained during the QR iterations. By the Hessenberg uniqueness theorem, we can organize the transformations in any way we like, as long as we come back to a Hessenberg form and still have the QR algorithm, only that we keep the same first column of the $Q$ factor. This is the rationale for implicit shift QR.

Let us follow this for a single shift $\sigma$. The first column of $A$ will now have only two elements $a_{1,1} - \sigma$ and $a_{2,1}$. Premultiply with an elementary rotation $P_{1,2}$ in the $(1, 2)$ to transform $a_{2,1}$ to zero. Now postmultiply with its transpose $P_{1,2}^T$ to complete the similarity transformation. This will introduce a nonzero at position $a_{3,1}$ outside the Hessenberg profile. This profile is restored by a process named chasing. Rotate in the $(2, 3)$ plane to zero $a_{3,1}$ but getting a new element in $a_{4,2}$. Continue and do $n - 2$ rotations until the extra element falls below row $n$.

1.3 Iterative eigenvalue algorithms

Iterative algorithms compute a sequence of vectors that hopefully converges to an eigenvector. The most basic iteration is the power method, where the $x_0$ is a starting guess and then a sequence $x_k$ is computed by

\[ x_k = A x_{k-1} \]
After many iterations \( x_k \) will tend to an eigenvector, corresponding to the eigenvalue \( \lambda_1 \) that is largest in absolute value, provided there is only one such eigenvalue.

\[
K_k(A, x_0) = \{ x_0, A x_0, \ldots, A^{k-1} x_0 \}
\]  
(1.6)

(Here the brackets \{ \ldots \} mean linear span of the columns given)

For any given vector that is supposed to approximate an eigenvector, we use the Rayleigh quotient to give an eigenvalue approximation.

**Definition 4 (Rayleigh Quotient)** For a given matrix pencil \((A, B)\) we define the Rayleigh quotient,

\[
\rho(x) = \frac{x^H A x}{x^H B x}
\]  
(1.7)

For a Hermitian pair \((A, B)\) with \(B\) positive definite, the eigenvalues are stationary points of the Rayleigh quotient. We have the Fischer theorem:

**Theorem 5 (Fischer)** Let the eigenvalues of \((A, B)\) be \( \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \). Then

\[
\lambda_i = \max_{\dim(x) = i, \; \forall \neq x \in X} \min \frac{x^H A x}{x^H B x}
\]  
(1.8)

We get the connection to the Krylov space when we note that it expands in the direction of the gradient of the Rayleigh quotient:

\[
g(x) = \rho'(x) = \frac{2}{x^H B x} (A x - \rho(x) B x)
\]  
(1.9)

### 1.3.1 Lanczos algorithm

### 1.3.2 Spectral Transformation
Bibliography


