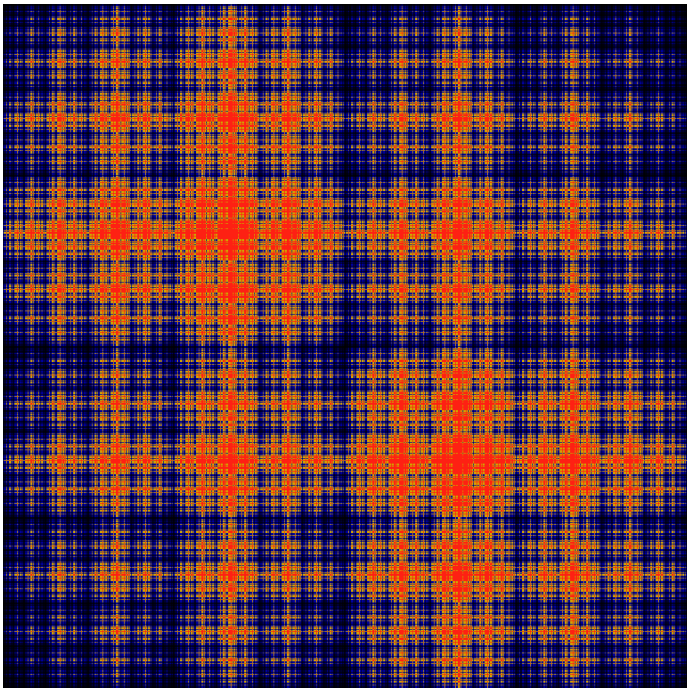
Applied Mathematics
and
Computing

Volume I



List of Contributors

J. Humpherys
Brigham Young University

J. Webb
Brigham Young University

R. Murray
Brigham Young University

J. West
University of Michigan

R. Grout
Brigham Young University

K. Finlinson
Brigham Young University

A. Zaitzeff
Brigham Young University

Preface

This lab manual is designed to accompany the textbook *Foundations of Applied Mathematics* by Dr. J. Humpherys.

Lab 15

Algorithms: Eigenvalue Solvers

Lesson Objective: *Implement the QR algorithm for finding eigenvalues.*

Eigenvalues are hard to find

Finding the eigenvalues of $n \times n$ matrix A means solving the following equation, where x is a nonzero vector and λ is a scalar.

$$\begin{aligned} Ax &= \lambda x \\ Ax - \lambda x &= 0 \\ (A - \lambda I)x &= 0 \end{aligned} \tag{15.1}$$

Since x is nonzero, (15.1) means $A - \lambda I$ must be singular. Thus $det(A - \lambda I) = 0$. This determinant is often notated $det(A-\lambda I) = p(\lambda)$ and is called the *characteristic polynomial* of A. The roots of the characteristic polynomial are the eigenvalues of A.

If A is $n \times n$, the degree of $p(\lambda) = n$. Finding the roots is easy for small n, but it becomes difficult or impossible as n increases. Abel's Theorem outlines the problem.

Theorem 15.1. Abel's Impossibility Theorem: *There is no general algebraic solution for solving a polynomial equation of degree $n > 4$.*

Therefore, there is no method that will exactly find the eigenvalues of an arbitrary matrix. This is a significant result. In practice it means that we often rely on iterative methods, which converge to the eigenvalues.

The QR algorithm

There are many such iterative methods for finding eigenvalues. We will explore one of the simplest: the QR algorithm. The following recurrence describes the QR Algorithm in its most basic form.

$$A_0 = A, \quad A_k = Q_k R_k, \quad A_{k+1} = R_k Q_k$$

where Q_k, R_k is the QR decomposition of A_k. Yes, it's as easy as it looks. All this algorithm does at each step is find the QR decomposition of A_k and multiply Q_k and R_k together again but in the opposite order. How does this simple algorithm find the eigenvalues of A?

$A_{k+1} \sim A_k$ (where $\sim$ denotes matrix similarity). Then $A_n \sim A$ for all n. This statement shows that A_n has the same eigenvalues as A. Preservation of eigenvalues is the first important feature that makes the algorithm work. The other important feature is that each iteration of the algorithm effectively transfers some of the"mass" from the lower to the upper triangle. Under very general conditions, A_n will converge to a matrix of the form

$$S = \begin{pmatrix} S_1 & * & \cdots & * \\ 0 & S_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & * \\ 0 & \cdots & 0 & S_m \end{pmatrix} \tag{15.2}$$

where S_i is either a 1×1 or a 2×2 matrix. For most matrices A, all the S_i will be 1×1, so S will be an upper triangular matrix. In this case, S is called the *Schur form* of A. The eigenvalues of A are on the main diagonal of S.

The only case where S is not upper triangular is when A is a real but not symmetric matrix. In this case, though A is real, it may have complex eigenvalues. These eigenvalues occur in complex conjugate pairs. Each of these pairs corresponds to a 2×2 block in S, where the eigenvalues of the 2×2 block are the complex conjugate pair of eigenvalues of A. In this case, S is called the *real Schur form* of A.

Hessenberg preconditioning

Recall from Lab **??** that an upper Hessenberg matrix looks like

$$\begin{pmatrix} * & * & * & \cdots & * \\ * & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & * & * \end{pmatrix}$$

and that every matrix is similar to an upper Hessenberg matrix. Hessenberg reduction also preserves eigenvalues. It is a good idea to reduce to Hessenberg form

before continuing with the QR algorithm. You'll converge to the Schur form faster this way, since Hessenberg matrices are already close to upper triangular.

Problem 1 Code the QR algorithm. Have your function accept an $n \times n$ matrix A and a number of iterations, and return all the eigenvalues of A. Note that you will need to find the eigenvalues of the 2×2 S_i directly, if there are any. Since your own implementations of QR decomposition and Hessenberg reduction may not handle complex matrices, you should use the ones in `scipy.linalg`.

Problem 2 Test your implementation with random matrices. Try real, complex, symmetric, and Hermitian matrices. Compare your output to the output from the eigenvalue solver. How many iterations are necessary? How large can A be?

The QR algorithm is not the only iterative method used to find eigenvalues. Arnoldi iteration is similar to the QR algorithm but exploits sparsity. Other methods include the Jacobi method and the Rayleigh quotient method.

It is important to remember that eigenvalue solvers can be wrong, particularly for matrices that are ill-conditioned.