Week 5

Numerical Diagonalization

1 Motivation: Diagonalization Ain't Easy

Suppose we want to find the eigenvalues and a corresponding set of linearly independent eigenvectors for the following matrix:

$$S = \begin{pmatrix} 2.2 & 15.9 & 6. & 17.2 & 18.4 \\ 15.9 & 31.4 & 11.1 & 10.1 & 11.9 \\ 6. & 11.1 & 14.6 & 8.4 & 44.1 \\ 17.2 & 10.1 & 8.4 & 5. & 7.6 \\ 18.4 & 11.9 & 44.1 & 7.6 & 12.2 \end{pmatrix} \tag{1}$$

Since the matrix is symmetric, the Spectral Theorem of linear algebra guarantees that this matrix is diagonalizable and that all of its eigenvalues are real. Knowing this, we might be tempted to compute the characteristic polynomial

$$\begin{aligned} \chi_S(\lambda) = -\lambda^5 + 65.4\lambda^4 + 1913.49\lambda^3 \\ - 68341.8\lambda^2 - 468342\lambda + 5775620 \times 10^6 \end{aligned} \tag{2}$$

and factor it with pencil and paper, but there are theorems of algebra, like the Abel-Ruffini theorem, implying that there is no simple, general formula to factor polynomials of degree 5 and higher. So what do we do? What if the matrix is much larger than 5×5?

Over the last couple of centuries, mathematicians have developed some powerful algorithms for computing the eigenvalues and eigenvectors of matrices to one's desired accuracy. We will study one of the oldest of these methods: the **Jacobi eigenvalue algorithm**. It is nowhere near the fastest method, but it is a robust method and relatively easy to program. Before we discuss the algorithm, we review some math and introduce some potentially new math that will help with understanding the algorithm.

2 Review of Some Linear Algebra

Suppose that S is an $n \times n$ real, symmetric matrix which means that its entries are real numbers, and its transpose equals itself;

$$S^T = S. \tag{3}$$

The Spectral Theorem of linear algebra guarantees that there is an orthogonal matrix O that diagonalizes S which means that there is a diagonal matrix D such that

$$O^{-1} S O = D. \tag{4}$$

The diagonal entries of D are the eigenvalues of S, which are guaranteed to be real since S is symmetric, and the columns of O are corresponding eigenvectors of S. Since O is orthogonal, its transpose equals its inverse;

$$O^{-1} = O^T, \tag{5}$$

and this means that we can also write

$$O^T S O = D. \tag{6}$$

Consider instead an $n \times n$ Hermitian matrix H which means that the entries of H can be complex numbers, and H equals its adjoint;

$$H^\dagger = H, \tag{7}$$

where the adjoint operation $\dagger$ is simply defined as taking the transpose followed by complex conjugation of all entries of the matrix;

$$H^\dagger = (H^T)^*. \tag{8}$$

In this case, the Spectral Theorem guarantees that there is a unitary matrix U that diagonalizes H which means that there is a diagonal matrix D such that

$$U^{-1} H U = D. \tag{9}$$

The diagonal entries of D are the eigenvalues of S, which are guaranteed to be real since S is hermitian, and the columns of U are corresponding eigenvectors of S. Since U is unitary, its adjoint equals its inverse;

$$U^\dagger = U^{-1}, \tag{10}$$

and this means that we can also write

$$U^\dagger H U = D. \tag{11}$$

3 Some probably new, but useful math

As in the last section, we take S to be a symmetric matrix. Suppose we want to find an orthogonal matrix O that diagonalizes S. What we will find numerically is that we can only ever find an O that approximately diagonalizes S to some desired precision, but what exactly does "approximately diagonal" mean? How do we measure how close to diagonal a matrix is? Here is one way: consider the following function that takes a complex matrix as its input and outputs a non-negative real number:

$$\text{off}(M) = \sqrt{\sum_{i \neq j} |M_{ij}|^2}. \tag{12}$$

This function is taking the norm off-diagonal elements of M, squaring each one, summing all of these squared elements, and then taking the square root. A nice way of thinking about this function is that if you were to arrange all of the off-diagonal elements of M as a single, long vector, and if you were to compute the norm of this vector, then you would get the same thing as off(M). If M is diagonal, off M is equal to 0, so one way of saying that M is "approximately" diagonal is to say that off(M) is close to 0. But there's actually a slightly better way to do this that is physically well-motivated.

The issue is that if the entries of M represent physical quantities, masses for example, then the size of the entries of M will depend on the system of units one uses, e,g, kilograms v. grams, so the concept of entries of M or off M being small will depend on the system of units. However, we can get around this in the following way. For any complex or real $n \times n$ matrix M, we define the **Frobenius norm** of the matrix as follows:

$$|M| = \sqrt{\sum_{i,j=1}^{n} |M_{ij}|^2}. \tag{13}$$

You can think of this norm as follows: if you were to unwrap all of the entries of M into a vector of length n^2, then $|M|$ gives the norm of that vector. This norm gives us a notion of the size of the matrix M and has the same units as the entries of M. We can now use this norm to give a unit-system-independent notion of smallness for off(M). We say that off(M) is small provided it is small compared to

$|M|$. In practice, if we were to do numerical computation, we could pick a small number ϵ as our tolerance and we would say that we are satisfied that M is approximately diagonal provided

$$\text{off}(M) < \epsilon |M|. \tag{14}$$

The question that remains is now:

> For a given symmetric or hermitian matrix, how can we algorithmically compute a similariy transformation that approximately diagonalizes it?

We begin to answer this in the next section via a concrete example that reveals the general idea behind the algorithm but hides some of the details.

4 A concrete 3×3 example

Consider the following 3×3 symmetric matrix:

$$S = \begin{pmatrix} 1 & 1.2 & 2 \\ 1.2 & 3 & 1.2 \\ 2 & 1.2 & 1 \end{pmatrix} \tag{15}$$

The idea behind the diagonalization algorithm we will study is to successively apply similarity transformations to S so that ultimately, all of the off-diagonal elements of A are as close to zero as desired, and S will therefore have been approximately diagonalized to one's desired accuracy. Each similarity transformation we apply to S will be of the form

$$J^{-1} S J \tag{16}$$

where J is a rotation matrix (an orthogonal matrix with positive determinant). Since J is a rotation, $J^{-1} = J^T$, so this similarlity transformation can also be written as $J^T S J$. We will scan through the off-diagonal elements of S, row-by-row, and apply a similarity transformation that successively attempts to turn that off-diagonal element into a zero. Let's see how this works explicitly. Note that for the remainder of this computation, we only display number to

hundreths place precision even though computations are being performed on computer to machine precision. Consider the matrix

$$J_1 = \begin{pmatrix} 0.91 & 0.42 & 0.00 \\ -0.42 & 0.91 & 0.00 \\ 0.00 & 0.00 & 1.00 \end{pmatrix} \tag{17}$$

If we apply a similarity transformation to S using this matrix, we get a new matrix S_1:

$$S_1 = J_1^T A J_1 = \begin{pmatrix} 0.44 & 0.00 & 1.3 \\ 0.00 & 3.56 & 1.94 \\ 1.30 & 1.94 & 1.00 \end{pmatrix} \tag{18}$$

Notice that the 12 element of the matrix has been set of zero by this rotation. Suppose now that we define

$$J_2 = \begin{pmatrix} 0.78 & 0.00 & 0.63 \\ 0.00 & 1.00 & 0.00 \\ -0.63 & 0.00 & 0.78 \end{pmatrix} \tag{19}$$

Again we apply a similarity transformation with J_2, but this time to the new matrix S_1, and we obtain

$$S_2 = J_2^T S_1 J_2 = \begin{pmatrix} -0.61 & -1.22 & 0.00 \\ -1.22 & 3.56 & 1.51 \\ 0.00 & 1.51 & 2.05 \end{pmatrix} \tag{20}$$

Notice that we have now set the 22 element of the matrix to zero by the rotation J_2. Let us do this one more time, except let us apply a similarity transformation to set the 23 element to zero. Define J_3 by

$$J_3 = \begin{pmatrix} 1.00 & 0.00 & 0.00 \\ 0.00 & 0.85 & 0.53 \\ 0.00 & 0.53 & 0.85 \end{pmatrix} \tag{21}$$

then

$$S_3 = J_3^T S_2 J_3 = \begin{pmatrix} -0.61 & -1.03 & 0.64 \\ -1.03 & 4.49 & 0.00 \\ 0.64 & 0.00 & 1.12 \end{pmatrix} \tag{22}$$

So at this point, you're probably thinking: hmmm so it seems that it's possible to set any given off-diagonal element of the matrix to zero by applying an appropriate similarity transformation, but to diagonalize the matrix, don't we want a similarity transformation that sets all of the off-diagonal elements to zero? Yes, and although it may not seem like it, the steps we have taken so far have gotten us further to accomplishing that goal. To see how this is so, we note that a bit of computation reveals the following:

$$\text{off}(S) = 5.25 \tag{23}$$
$$\text{off}(S_1) = 4.66 \tag{24}$$
$$\text{off}(S_2) = 3.87 \tag{25}$$
$$\text{off}(S_3) = 2.43. \tag{26}$$

Although our steps in computing S_1, S_2, S_3 do not diagonalize S, they do get closer to doing so – the matrix is becoming closer to approximately diagonal. In fact, if we apply three more similarity transformations with similarly defined rotation matrices J_4, J_5, J_6, J_7, then we obtain a sequence of matrices S_4, S_5, S_6, S_7 that become closer and closer to diagonal. In fact, S_7 is

$$S_7 = J_7^T S_6 J_7 = \begin{pmatrix} -1.00 & 0.00 & 0.00 \\ 0.00 & 4.70 & 0.00 \\ 0.00 & 0.00 & 1.30 \end{pmatrix} \tag{27}$$

So by successively applying 7 similarity transformations to S, we are able to reduce all of its off-diagonal elements to zero with a precision of two decimal places. In fact, a bit more computation reveals that to that precision,

$$\text{off}(S_4) = 1.28 \tag{28}$$
$$\text{off}(S_5) = 0.24 \tag{29}$$
$$\text{off}(S_6) = 0.07 \tag{30}$$
$$\text{off}(S_7) = 0.00. \tag{31}$$

In summary, we found that the matrix

$$S_7 = J_7^T(J_6^T(J_5^T(J_4^T(J_3^T(J_2^T(J_1^T S J_1)J_2)J_3)J_4)J_5)J_6)J_7 \tag{32}$$

is close to diagonal. This implies that if we define

$$R = J_1 J_2 J_3 J_4 J_5 J_6 J_6, \tag{33}$$

then

$$S_7 = R^T A R \tag{34}$$

is close to diagonal. This shows that we have found a similarity transformation that approximately diagonalizes A. The columns of R are therefore a set of linearly independent eigenvectors of A.

What we have just done is a concrete example of using the **Jacobi algorithm** to approximately diagonalize a matrix. However, the real trick behind this method is computation of the matrices $J_1, J_2, J_3, \ldots$ which we just pulled out of a hat in this section. At the end of these notes, we present general, detailed algorithms that reveal how these matrices are computed and formalize the Jacobi algorithm as a whole.

5 Adapting the algorithm to Hermitian matrices

The Jacobi algorithm, as its presented in the last section, allows one to diagonalize a real, symmetric matrix to within some desired tolerance, but what about if one wants to diagonalize a complex, hermitian matrix instead? It turns out that there is a clever way to simply adapt the real, symmetric algorithm to the complex, hermitian case.

Suppose that H is a hermitian matrix, then there exist unique real matrices S and A, symmetric and antisymmetric respectively, such that

$$H = S + iA. \tag{35}$$

Using the matrices S and A, form the following "augmented matrix" O which is symmetric but is $2n \times 2n$ instead of $n \times n$:

$$O = \begin{pmatrix} S & -A \\ A & S \end{pmatrix}. \tag{36}$$

Use the Jacobi algorithm to compute the rotation R that diagonalizes O and whose columns are the eigenvectors of O along with the diagonalized version $D = R^T O R$ of O. What you will now find is

that D has the form

$$D = \begin{pmatrix} \lambda_1 & & & & & & \\ & \lambda_1 & & & & & \\ & & \lambda_2 & & & & \\ & & & \lambda_2 & & & \\ & & & & \ddots & & \\ & & & & & \lambda_n & \\ & & & & & & \lambda_n \end{pmatrix} \tag{37}$$

where $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the eigenvalues of H, so they appear as redundant pairs as eigenvalues of the augmented matrix O. Therefore, one simply needs to extract only every other element of D to obtain the eigenvalues of H, as desired. As for the corresponding eigenvectors, each of the columns of R is of the form

$$\begin{pmatrix} u \\ v \end{pmatrix} \tag{38}$$

where u and v are $n \times 1$ column vectors. Each such column corresponds to one of the eigenvalues λ_k of O. To obtain the corresponding eigenvector of H, simply compute the sum

$$u + iv. \tag{39}$$

By extracting every other column of R and adding its upper-half, u, to i times its lower-half v, one obtains a basis of eigenvectors of H, as desired.

6 Algorithm Details

Given an $n \times n$ real, symmetric matrix S and a pair of integers (p, q) with $1 \leq p < q \leq n$, the following algorithm computes a rotation J such that the qp and pq elements of $J^T S J$ are zero:

Algorithm 1 Compute Jacobi Rotation

if $S_{pq} \neq 0$ **then**
  $\tau = (A_{qq} - A_{pp})/(2A_{pq})$
  if $\tau \geq 0$ **then**
    $t = 1/(\tau + \sqrt{1 + \tau^2})$
  else
    $t = 1/(\tau - \sqrt{1 + \tau^2})$
  end if
  $c = 1/\sqrt{1 + t^2}$
  $s = tc$
else
  $c = 1$
  $s = 0$
end if
$J = I_N$ {I is the $n \times n$ identity matrix}
$J_{pp} = c$
$J_{pq} = -s$
$J_{qp} = s$
$J_{qq} = c$

For a given symmetric matrix S, let $J_{S,p,q}$ denote the Jacobi rotation produced by 1. The following algorithm diagonalizes a given symmetric matrix S by cyclically scanning through rows and applying corresponding Jacobi rotations. The result is a that for a given tolerance ϵ, S is overwritten with a rotation R and a nearly diagonalized matrix $R^T SR$ satisfying $\text{off}(R^T SR) \leq \epsilon |A|$.

Algorithm 2 Cyclic Jacobi Diagonalization

$R = I_n$ {I is the $n \times n$ identity matrix}
$\delta = \epsilon |S|$
while $\text{off}(S) > \delta$ **do**
  for $p = 1$ to $n - 1$ **do**
    for $q = p + 1$ to n **do**
      $S = J_{S,p,q}^T S J_{S,p,q}$
      $R = R J_{S,p,q}$
    end for
  end for
end while