Lecture 9 Notes

1 Eigenvalues and Eigenvectors

In the last lecture, we discussed a class of iterative methods for solving the system $A\mathbf{x} = \mathbf{b}$. In particular, we found an equation for the error of our estimated solutions:

$$\mathbf{e}_k = M^k \mathbf{e}_0.$$

We saw before that this equation is straightforward if M and the $\mathbf{e}$'s are just numbers, but becomes more complicated when M is a matrix and the $\mathbf{e}$'s are vectors. We would like to use a similar method with the matrix version, but to do so we need to know when and how we are allowed to treat matrix multiplication like scalar multiplication. In particular, we would like to know when we are allowed to write

$$M\mathbf{x} = \lambda \mathbf{x}, \tag{1}$$

where M is a matrix, $\mathbf{x}$ is a vector and λ is a number.

The notation here is a bit confusing. We are ultimately interested in using the matrix M from last lecture in equation (1), but the equation is perfectly valid for *any* square matrix, not just that particular one. In addition, the vector $\mathbf{x}$ in (1) has nothing to do with the solution to our system $A\mathbf{x} = \mathbf{b}$. You should think of equation (1) as follows: You are given some fixed matrix M (it can be *any* square matrix) and you are looking for a vector $\mathbf{x}$ and a number λ such that the equation holds.

The number λ in equation (1) is called an *eigenvalue* of M and the vector $\mathbf{x}$ is called an *eigenvector* of M corresponding to λ.

Let's look at a few simple examples. Let

$$M = \begin{pmatrix} 1 & 2 \\ 0 & 3 \end{pmatrix} \text{ and } \mathbf{x} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

We have

$$M\mathbf{x} = \begin{pmatrix} 1 & 2 \\ 0 & 3 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} = 3\mathbf{x}.$$

This means that 3 is an eigenvalue of M with corresponding eigenvector $(1, 1)^T$. (Remember that the T stands for transpose and just means "make this a column vector.")

Similarly, if we let

$$\mathbf{x} = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

then

$$M\mathbf{x} = \begin{pmatrix} 1 & 2 \\ 0 & 3 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}.$$

This cannot possibly be written as $\lambda\mathbf{x}$ because any multiple of $\mathbf{x}$ would have a 0 in the first entry. That means that $(0, 1)^T$ is *not* an eigenvector of M.

It turns out (and you should experiment until you are convinced that this is true) that most vectors are not eigenvectors of M. The only possible eigenvectors are multiples of $(1, 1)^T$ and multiples of $(1, 0)^T$. (If $\mathbf{x}$ is an eigenvector, then so is any multiple of $\mathbf{x}$. We therefore usually ignore these constant multiples and just say that M has two different eigenvectors.) Moreover, there are only two eigenvalues of M. We already saw that 3 is an eigenvalue of M, and it is also easy to check that 1 is an eigenvalue of M. This is not a coincidence; it follows from the following theorem:

Every $n \times n$ matrix M has n eigenvalues. Almost[1] every $n \times n$ matrix M has n different eigenvectors.

The course videos show how to find eigenvalues and eigenvectors of a 2×2 matrix by hand. It is also possible to use that method for 3×3 matrices (but it is much more difficult). For anything larger than a 3×3 matrix it is not usually possible to find eigenvalues/eigenvectors by hand. Since we are primarily interested in large matrices. we will therefore not worry about finding these values by hand. Instead, we will let Matlab do the hard work for us.

In Matlab, we can find the eigenvalues and eigenvectors of a matrix M with the `eig` command. This command has many different modes (and you should, as always, investigate the help file to learn more about it) but we will only use two. If you ask for one output, such as `E = eig(M)`, then Matlab returns a vector with all n eigenvalues of M. If you ask for two outputs, such as `[V, D] = eig(M)`, then each column of the matrix V will be an eigenvector of M and each diagonal entry of D will be the corresponding eigenvalue. Matlab doesn't just use this format because it's a convenient way to store all of the eigenvectors; the matrices V and D have important mathematical significance. It turns out[2] that we can rewrite the matrix M using V and D as follows:

$$M = VDV^{-1}. \tag{2}$$

This is called an *eigendecomposition* of M.

[1]It is possible for several of the eigenvalues to be the same, just like how a quadratic equation always has two solutions but they might coincide. If two or more eigenvalues are the same then it is possible for them to share the same eigenvector or to have different eigenvectors. We will not worry about this problem in our class.

[2]We will verify this in Matlab, but not prove it.

It is worth noting that the eigenvectors and eigenvalues of a matrix might be complex. That is, they might involve imaginary numbers. This is not a problem, since we are only ever worried about the magnitude of these eigenvalues, which is always real.

The eigendecomposition is particularly useful for computing powers of M. To see this, notice that

$$M^2 = \left(VDV^{-1}\right)\left(VDV^{-1}\right) = VDDV^{-1} = VD^2V^{-1},$$

because the innermost V and V^{-1} cancel out. Similarly,

$$M^3 = M^2M = \left(VD^2V^{-1}\right)\left(VDV^{-1}\right) = VD^2V^{-1}VDV^{-1} = VD^2DV^{-1} = VD^3V^{-1}.$$

The pattern should quickly become apparent. In general, we have

$$M^k = VD^kV^{-1}.$$

The reason this is so convenient is that D is a diagonal matrix, and it is easy to compute powers of a diagonal matrix: We just raise each entry of the matrix to the same power. For instance,

$$\begin{pmatrix} 2 & 0 \\ 0 & 1/2 \end{pmatrix}^k = \begin{pmatrix} 2^k & 0 \\ 0 & (1/2)^k \end{pmatrix}.$$

2 Convergence of Matrix Splitting Algorithms

We are now ready to determine when our matrix splitting algorithms converge. Remember that the error of our guesses is governed by the equation

$$\mathbf{e}_k = M^k\mathbf{e}_0.$$

If we rewrite this using the eigendecomposition of M (which we can calculate using `eig`), we get

$$\mathbf{e}_k = VD^kV^{-1}\mathbf{e}_0.$$

Remember that D is a diagonal matrix whose entries are the eigenvalues of M, and therefore the entries of D^k are the eigenvalues of M raised to the kth power. If *any* of the eigenvalues of M are bigger than 1, then one of entries of D^k will grow larger and larger as k increases. This means that D^k will have some entry that goes off to infinity. When we multiply D^k by V and V^{-1} and e_0, there still be at least one extremely large entry, so the error will still be very large. On the other hand, if *all* of the eigenvalues of M are smaller than one, then the entries of D^k will get smaller and smaller as k increases. For large values of k, the matrix D will be almost all zeros, which means that $VDV^{-1}\mathbf{e}_0$ will also be almost all zeros, so the error will get very small. This gives us our final rule:

- If *any* of the eigenvalues of M are larger than one (in magnitude), then $\mathbf{e}_k$ will go to infinity as k increases.
- If *all* of the eigenvalues of M are smaller than one (in magnitude), then $\mathbf{e}_k$ will go to zero as k increases.

This gives us an easy way to check if our matrix splitting method will converge. We compute P, T and $M = -P^{-1}T$, then use the `eig` command to compute the eigenvalues of M. If the largest (in magnitude) of these eigenvalues is less than 1, then we know that our method will converge and we can proceed with the algorithm. If any of the eigenvalues are greater than one, then we know that our method will fail and we can try a different method (or choose a different splitting).